

iStock 智能股票管理系统 - 部署文档

目录

- [部署概述](#)
- [环境准备](#)
- [前端部署](#)
- [后端部署](#)
- [数据库部署](#)
- [Nginx配置](#)
- [SSL证书配置](#)
- [监控与维护](#)
- [故障处理](#)

部署概述

本文档详细说明iStock智能股票管理系统的完整部署流程，包括前端、后端、数据库和Nginx反向代理的部署配置。

部署架构



系统要求

硬件要求

组件	最低配置	推荐配置
CPU	2核	4核+
内存	4GB	8GB+
存储	50GB SSD	100GB+ SSD
网络	10Mbps	100Mbps+

软件要求

软件	版本要求	说明
操作系统	Ubuntu 20.04+ / CentOS 8+	Linux发行版
Nginx	1.18+	Web服务器
Node.js	18.0+	JavaScript运行时
MySQL	8.0+	数据库服务器
PM2	5.0+	Node.js进程管理

环境准备

1. 系统更新

```
# Ubuntu/Debian
sudo apt update && sudo apt upgrade -y

# CentOS/RHEL
sudo yum update -y
```

2. 安装基础工具

```
# Ubuntu/Debian
sudo apt install -y curl wget git unzip htop vim

# CentOS/RHEL
sudo yum install -y curl wget git unzip htop vim
```

3. 创建应用用户

```
# 创建专用用户
sudo useradd -m -s /bin/bash istock
sudo usermod -aG sudo istock

# 切换到应用用户
sudo su - istock
```

4. 创建目录结构

```
# 创建应用目录
sudo mkdir -p /var/www/istock
sudo mkdir -p /var/log/istock
sudo mkdir -p /etc/istock

# 设置权限
sudo chown -R istock:istock /var/www/istock
sudo chown -R istock:istock /var/log/istock
sudo chown -R istock:istock /etc/istock
```

1. 获取源代码

```
# 克隆代码仓库
cd /var/www/istock
git clone <your-repository-url> .

# 或者上传代码包
scp -r ./SMMS_WEB istock@server:/var/www/istock/
```

2. 安装Node.js

```
# 使用NodeSource安装Node.js 18
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# 验证安装
node --version
npm --version
```

3. 构建前端项目

```
# 进入前端目录
cd /var/www/istock

# 安装依赖
npm install

# 构建生产版本
npm run build

# 验证构建结果
ls -la dist/
```

4. 配置环境变量

```
# 创建生产环境配置
cat > .env.production << EOF
NODE_ENV=production
VITE_API_BASE_URL=https://api.istock.com
VITE_APP_TITLE=iStock智能股票管理系统
EOF
```

5. 优化构建

```
# 使用生产环境配置构建
npm run build

# 压缩静态文件
npm install -g gzip-cli
cd dist
gzip -k -r ./\*
cd ..
```



后端部署

1. 安装PM2进程管理器

```
# 全局安装PM2
sudo npm install -g pm2

# 验证安装
pm2 --version
```

2. 配置后端环境

```
# 进入后端目录
cd /var/www/istock/server

# 复制环境配置模板
cp .env.example .env

# 编辑生产环境配置
vim .env
```

3. 生产环境配置

```
# 生产环境 .env 配置
cat > .env << EOF
# 数据库配置
DB_HOST=localhost
DB_PORT=3306
DB_USER=istock_user
DB_PASSWORD=your_secure_password
DB_NAME=stock_app

# JWT配置
JWT_SECRET=your-super-secret-jwt-key-change-in-production-very-long-random-string
JWT_EXPIRE=7d

# 服务器配置
PORT=3001
NODE_ENV=production

# 外部API配置
```

```
MAIRUI_LICENSE=LICENCE-66D8-9F96-0C7F0FBCD073
```

```
# 日志配置
LOG_LEVEL=info
LOG_FILE=/var/log/istock/app.log
EOF
```

4. 安装后端依赖

```
# 安装依赖
cd /var/www/istock/server
npm install --production

# 验证安装
npm list --depth=0
```

5. 数据库初始化

```
# 创建数据库（如不存在）
mysql -u root -p << EOF
CREATE DATABASE IF NOT EXISTS stock_app CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
CREATE USER IF NOT EXISTS 'istock_user'@'localhost' IDENTIFIED BY
'your_secure_password';
GRANT ALL PRIVILEGES ON stock_app.* TO 'istock_user'@'localhost';
FLUSH PRIVILEGES;
EOF

# 运行数据库初始化脚本
node -e "
const mysql = require('mysql2/promise');
require('dotenv').config();

async function initDB() {
  const connection = await mysql.createConnection({
    host: process.env.DB_HOST,
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    database: process.env.DB_NAME
  });

  // 创建用户表
  await connection.execute(`
    CREATE TABLE IF NOT EXISTS users (
      id INT AUTO_INCREMENT PRIMARY KEY,
      username VARCHAR(50) UNIQUE NOT NULL,
      email VARCHAR(100) UNIQUE NOT NULL,
      password_hash VARCHAR(255) NOT NULL,
      created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
      updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    )
  `);

  // 创建股票表
```

```

await connection.execute(\`
  CREATE TABLE IF NOT EXISTS stocks (
    id INT AUTO_INCREMENT PRIMARY KEY,
    stock_code VARCHAR(10) UNIQUE NOT NULL,
    stock_name VARCHAR(100) NOT NULL,
    exchange VARCHAR(20) NOT NULL,
    current_price DECIMAL(10,2),
    change_percent DECIMAL(8,4),
    market VARCHAR(10) NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
  )
\`);

// 创建收藏表
await connection.execute(\`
  CREATE TABLE IF NOT EXISTS favorites (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    stock_code VARCHAR(10) NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    UNIQUE KEY unique_user_stock (user_id, stock_code)
  )
\`);

console.log('Database initialized successfully');
await connection.end();
}

initDB().catch(console.error);
"
```

6. PM2配置文件

```

# 创建PM2配置文件
cat > ecosystem.config.js << EOF
module.exports = {
  apps: [{
    name: 'istock-server',
    script: './server2.0.js',
    cwd: '/var/www/istock/server',
    instances: 'max',
    exec_mode: 'cluster',
    env: {
      NODE_ENV: 'production',
      PORT: 3001
    },
    error_file: '/var/log/istock/error.log',
    out_file: '/var/log/istock/out.log',
    log_file: '/var/log/istock/combined.log',
    time: true,
    max_memory_restart: '1G',
    node_args: '--max-old-space-size=1024'
  ]
}
```

```
}]
};
EOF
```

7. 启动后端服务

```
# 启动服务
cd /var/www/istock/server
pm2 start ecosystem.config.js

# 保存PM2配置
pm2 save

# 设置开机自启
pm2 startup | sudo bash
sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -u istock --hp /home/istock
sudo systemctl enable pm2-istock

# 验证服务状态
pm2 status
pm2 logs istock-server
```

数据库部署

1. 安装MySQL

```
# Ubuntu安装MySQL 8.0
sudo apt update
sudo apt install -y mysql-server mysql-client

# CentOS安装MySQL 8.0
sudo yum install -y mysql-server mysql-client

# 启动MySQL服务
sudo systemctl start mysql
sudo systemctl enable mysql
```

2. MySQL安全配置

```
# 运行安全配置脚本
sudo mysql_secure_installation

# 配置选项：
# - 设置root密码
# - 移除匿名用户
# - 禁止root远程登录
# - 移除测试数据库
# - 重新加载权限表
```

3. 数据库优化配置

```
# 编辑MySQL配置
sudo vim /etc/mysql/mysql.conf.d/mysqld.cnf

# 添加优化配置
[mysqld]
# 基础配置
bind-address = 127.0.0.1
port = 3306
datadir = /var/lib/mysql
socket = /var/run/mysqld/mysqld.sock

# 性能优化
innodb_buffer_pool_size = 1G
innodb_log_file_size = 256M
innodb_flush_log_at_trx_commit = 2
max_connections = 200
query_cache_size = 64M

# 字符集配置
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci

# 重启MySQL
sudo systemctl restart mysql
```

4. 数据库备份配置

```
# 创建备份脚本
cat > /etc/istock/backup.sh << EOF
#!/bin/bash

BACKUP_DIR="/var/backups/istock"
DB_NAME="stock_app"
DB_USER="istock_user"
DB_PASS="your_secure_password"
DATE=$(date +%Y%m%d_%H%M%S)

# 创建备份目录
mkdir -p \${BACKUP_DIR}

# 备份数据库
mysqldump -u \${DB_USER} -p\${DB_PASS} \${DB_NAME} | gzip >
\${BACKUP_DIR}/istock_backup_\${DATE}.sql.gz

# 删除7天前的备份
find \${BACKUP_DIR} -name "istock_backup_*.sql.gz" -mtime +7 -delete

echo "Database backup completed: \${BACKUP_DIR}/istock_backup_\${DATE}.sql.gz"
EOF

# 设置执行权限
chmod +x /etc/istock/backup.sh
```



```
# 添加到crontab（每日凌晨2点备份）
echo "0 2 * * * /etc/istock/backup.sh" | crontab -
```

Nginx配置

1. 安装Nginx

```
# Ubuntu安装Nginx
sudo apt update
sudo apt install -y nginx

# CentOS安装Nginx
sudo yum install -y nginx

# 启动并启用Nginx
sudo systemctl start nginx
sudo systemctl enable nginx
```

2. 创建Nginx配置文件

```
# 创建站点配置
sudo vim /etc/nginx/sites-available/istock
```

```
# istock Nginx配置
server {
    listen 80;
    server_name istock.com www.istock.com;

    # 重定向到HTTPS
    return 301 https://\${server_name}\$request_uri;
}

server {
    listen 443 ssl http2;
    server_name istock.com www.istock.com;

    # SSL配置
    ssl_certificate /etc/ssl/certs/istock.com.crt;
    ssl_certificate_key /etc/ssl/private/istock.com.key;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    # 安全头部
    add_header X-Frame-Options DENY;
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains"
    always;
```

```

# 前端静态文件
location / {
    root /var/www/istock/dist;
    index index.html;
    try_files $uri $uri/ /index.html;

    # 静态资源缓存
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2|ttf|eot)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
        gzip_static on;
    }
}

# API代理
location /api/ {
    proxy_pass http://127.0.0.1:3001;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    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;
    proxy_cache_bypass $http_upgrade;
    proxy_read_timeout 300s;
    proxy_connect_timeout 75s;
}

# 日志配置
access_log /var/log/nginx/istock_access.log;
error_log /var/log/nginx/istock_error.log;

# Gzip压缩
gzip on;
gzip_vary on;
gzip_min_length 1024;
gzip_proxied any;
gzip_comp_level 6;
gzip_types
    text/plain
    text/css
    text/xml
    text/javascript
    application/json
    application/javascript
    application/xml+rss
    application/atom+xml
    image/svg+xml;
}

```

3. 启用站点配置

```
# 启用站点
sudo ln -s /etc/nginx/sites-available/istock /etc/nginx/sites-enabled/

# 删除默认站点
sudo rm -f /etc/nginx/sites-enabled/default

# 测试配置
sudo nginx -t

# 重新加载配置
sudo systemctl reload nginx
```

4. Nginx性能优化

```
# 编辑主配置文件
sudo vim /etc/nginx/nginx.conf

# 优化配置
worker_processes auto;
worker_connections 2048;

# 启用HTTP/2
listen 443 ssl http2;

# 限制请求大小
client_max_body_size 10M;

# 连接超时
keepalive_timeout 65;
client_body_timeout 12;
client_header_timeout 12;
send_timeout 10;
```

SSL证书配置

1. 使用Let's Encrypt免费证书

```
# 安装Certbot
sudo apt install -y certbot python3-certbot-nginx

# 获取SSL证书
sudo certbot --nginx -d istock.com -d www.istock.com

# 配置自动续期
echo "0 12 * * * /usr/bin/certbot renew --quiet" | sudo crontab -
```

2. 手动配置证书（如果有商业证书）

```
# 创建SSL目录
sudo mkdir -p /etc/ssl/certs /etc/ssl/private

# 复制证书文件
sudo cp istock.com.crt /etc/ssl/certs/
sudo cp istock.com.key /etc/ssl/private/

# 设置权限
sudo chmod 644 /etc/ssl/certs/istock.com.crt
sudo chmod 600 /etc/ssl/private/istock.com.key

# 验证证书
sudo openssl x509 -in /etc/ssl/certs/istock.com.crt -text -noout
```

3. SSL配置优化

```
# 在Nginx配置中添加
ssl_session_timeout 1d;
ssl_session_cache shared:SSL:50m;
ssl_stapling on;
ssl_stapling_verify on;

# 启用OCSP
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
```



监控与维护

1. 系统监控

```
# 安装监控工具
sudo apt install -y htop iotop nethogs

# 创建系统监控脚本
cat > /etc/istock/monitor.sh << EOF
#!/bin/bash

# 系统资源监控
echo "=== System Status ==="
echo "CPU: \$(top -bn1 | grep "Cpu(s)" | awk '{print \$2}' | cut -d'%' -f1)"
echo "Memory: \$(free -m | awk 'NR==2{printf "%.2f%%", \$3*100/\$2}')"
echo "Disk: \$(df -h / | awk 'NR==2 {print \$5}')"

# 服务状态
echo -e "\n=== Service Status ==="
echo "Nginx: \$(systemctl is-active nginx)"
echo "MySQL: \$(systemctl is-active mysql)"
echo "PM2: \$(pm2 jlist | jq length) apps running"

# 应用日志
echo -e "\n=== Recent Errors ==="
```

```
tail -10 /var/log/istock/error.log 2>/dev/null || echo "No error logs"
EOF

chmod +x /etc/istock/monitor.sh
```

2. 日志管理

```
# 创建日志轮转配置
sudo vim /etc/logrotate.d/istock

# 配置内容
/var/log/istock/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 istock istock
    postrotate
        pm2 reloadLogs
    endscript
}

/var/log/nginx/istock_*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 www-data adm
    postrotate
        systemctl reload nginx
    endscript
}
```

3. 性能监控

```
# 安装Node.js监控模块
cd /var/www/istock/server
npm install pm2-logrotate

# 配置日志轮转
pm2 install pm2-logrotate
pm2 set pm2-logrotate:max_size 10M
pm2 set pm2-logrotate:retain 30
pm2 set pm2-logrotate:compress true
```

4. 健康检查脚本

```
# 创建健康检查脚本
cat > /etc/istock/health_check.sh << EOF
#!/bin/bash

# 检查HTTP状态
HTTP_STATUS=$(curl -s -o /dev/null -w "%{http_code}" https://istock.com)
if [ \${HTTP_STATUS} -ne 200 ]; then
    echo "HTTP check failed: \${HTTP_STATUS}"
    # 发送告警邮件或其他通知
fi

# 检查API响应
API_STATUS=$(curl -s -o /dev/null -w "%{http_code}"
https://istock.com/api/health)
if [ \${API_STATUS} -ne 200 ]; then
    echo "API check failed: \${API_STATUS}"
    # 重启应用服务
    pm2 restart istock-server
fi

# 检查数据库连接
DB_STATUS=$(mysql -u istock_user -p\${DB_PASS} -e "SELECT 1" stock_app
2>/dev/null; echo \${?})
if [ \${DB_STATUS} -ne 0 ]; then
    echo "Database check failed"
    # 发送告警
fi
EOF

chmod +x /etc/istock/health_check.sh

# 添加到crontab（每5分钟检查一次）
echo "*/5 * * * * /etc/istock/health_check.sh" | crontab -
```

故障处理

1. 常见问题及解决方案

前端问题

```
# 页面404错误
# 检查文件权限
sudo chown -R istock:istock /var/www/istock/dist
sudo chmod -R 755 /var/www/istock/dist

# 静态资源加载失败
# 检查Nginx配置
sudo nginx -t
sudo systemctl reload nginx

# 检查文件是否存在
```

```
ls -la /var/www/istock/dist/
```

后端问题

```
# 服务无法启动
# 检查PM2状态
pm2 status
pm2 logs istock-server

# 重启服务
pm2 restart istock-server

# 检查端口占用
sudo netstat -tlnp | grep 3001

# 检查环境变量
pm2 env 0
```

数据库问题

```
# 数据库连接失败
# 检查MySQL状态
sudo systemctl status mysql

# 检查数据库用户权限
mysql -u root -p
SHOW GRANTS FOR 'istock_user'@'localhost';

# 检查数据库是否存在
SHOW DATABASES LIKE 'stock_app';

# 修复数据库表
mysqlcheck -u istock_user -p stock_app --repair
```

Nginx问题

```
# 502 Bad Gateway错误
# 检查后端服务状态
curl http://127.0.0.1:3001/api/health

# 检查Nginx错误日志
sudo tail -f /var/log/nginx/istock_error.log

# 重启Nginx
sudo systemctl restart nginx
```

2. 紧急恢复流程

服务完全宕机

```
# 1. 检查系统状态
sudo systemctl status nginx mysql
pm2 status

# 2. 重启所有服务
sudo systemctl restart nginx mysql
pm2 restart all

# 3. 检查日志
sudo journalctl -u nginx -f
pm2 logs

# 4. 验证服务
curl -I https://istock.com
curl -I https://istock.com/api/health
```

数据库损坏

```
# 1. 停止应用服务
pm2 stop istock-server

# 2. 恢复最近的备份
cd /var/backups/istock
LATEST_BACKUP=$(ls -t istock_backup_*.sql.gz | head -1)
gunzip < \$LATEST_BACKUP | mysql -u istock_user -p stock_app

# 3. 重启应用服务
pm2 start istock-server
```

SSL证书过期

```
# 手动续期证书
sudo certbot renew --force-renewal

# 重启Nginx
sudo systemctl reload nginx

# 验证证书
openssl s_client -connect istock.com:443 -servername istock.com
```

3. 性能优化建议

数据库优化

```
-- 创建索引
CREATE INDEX idx_favorites_user_id ON favorites(user_id);
CREATE INDEX idx_stocks_stock_code ON stocks(stock_code);
CREATE INDEX idx_stock_data_code_date ON stock_data(stock_code, trade_date);
```



```
-- 分析查询性能
EXPLAIN SELECT * FROM favorites WHERE user_id = 1;
```

缓存配置

```
# 在Nginx中配置API缓存
location ~* /api/stocks/list$ {
    proxy_pass http://127.0.0.1:3001;
    proxy_cache api_cache;
    proxy_cache_valid 200 5m;
    proxy_cache_key "$scheme$request_method$host$request_uri";
}
```

应用优化

```
// 在server2.0.js中添加连接池配置
const pool = mysql.createPool({
  connectionLimit: 20,
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME
});
```
