ASMC API - Deployment Guide

Complete guide for deploying the ASMC API from development to production environments.

Table of Contents

Deployment Overview

Deployment Environments

Environment	URL	Purpose	Configuration
Development	`http://localhost:7055`	Local development	`.env.development`
Staging	`https://staging-api.asmcdae.in`	Testing	`.env.staging`
Production	`https://api.asmcdae.in`	Live system	`.env.production`

Deployment Checklist

- [ ] Server prepared (Ubuntu 20.04+)
- [ ] Node.js 18+ installed
- [ ] MongoDB 6.0+ installed and configured
- [ ] Application deployed
- [ ] Environment variables configured
- [ ] Nginx configured
- [ ] SSL certificate installed
- [ ] PM2 configured
- [ ] Monitoring set up
- [ ] Backup strategy implemented

Ubuntu Server Setup

Step 1: Server Preparation

```
# Update system packages
sudo apt update && sudo apt upgrade -y
```

```
# Install essential packages
sudo apt install -y curl wget git unzip software-properties-common apt-transport-https
ca-certificates gnupg lsb-release

# Create application user
sudo adduser asmc
sudo usermod -aG sudo asmc
su - asmc
```

Step 2: Install Node.js

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

# Verify installation
node --version  # Should show v18.x.x
npm --version   # Should show 9.x.x

# Install PM2 globally
sudo npm install -g pm2
```

Step 3: Install MongoDB

```
# Import MongoDB public GPG key
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -

# Add MongoDB repository
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-
org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

# Update package database
sudo apt-get update

# Install MongoDB
sudo apt-get install -y mongodb-org

# Start and enable MongoDB
sudo systemctl start mongod
sudo systemctl enable mongod

# Verify MongoDB is running
sudo systemctl status mongod
```

Step 4: Install Nginx

```
# Install Nginx
sudo apt install -y nginx

# Start and enable Nginx
sudo systemctl start nginx
```

```
sudo systemctl enable nginx

# Verify Nginx is running
sudo systemctl status nginx
```

Application Deployment

Step 1: Deploy Application

```
# Create application directory
sudo mkdir -p /opt/asmc-api
sudo chown asmc:asmc /opt/asmc-api

# Clone repository
cd /opt/asmc-api
git clone <repository-url> .

# Install dependencies
npm ci --production

# Set up environment
cp .env.example .env.production
nano .env.production
```

Step 2: Production Environment Configuration

```
# .env.production
PORT=7055
NODE_ENV=production

# Database
MONGO_URI=mongodb://localhost:27017/asmc_prod

# Authentication
JWT_SECRET=your-super-secure-jwt-secret-minimum-64-characters-long-for-production
JWT_EXPIRE=7d

# Email Configuration
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=your-production-email@gmail.com
SMTP_PASS=your-app-password
MSG91_AUTH_KEY=your-msg91-auth-key
MSG91_SENDER_ID=ASMC

# Image Processing
IMAGEKIT_PUBLIC_KEY=your-imagekit-public-key
IMAGEKIT_PRIVATE_KEY=your-imagekit-private-key
IMAGEKIT_URL_ENDPOINT=https://ik.imagekit.io/your-imagekit-id

# Payment Gateway
```

```
CCAVENUE_MERCHANT_ID=your-merchant-id
CCAVENUE_ACCESS_CODE=your-access-code
CCAVENUE_WORKING_KEY=your-working-key

# Biometric Integration
BIOMETRIC_IP=192.168.1.100
BIOMETRIC_PORT=4370

# Security
CORS_ORIGINS=https://asmcdae.in,https://admin.asmcdae.in
```

Step 3: Create PM2 Ecosystem File

```
# Create ecosystem configuration
cat > ecosystem.config.js << 'EOF'
module.exports = {
  apps: [{
    name: 'asmc-api',
    script: 'app.js',
    instances: 'max',
    exec_mode: 'cluster',
    env: {
      NODE_ENV: 'production',
      PORT: 7055
    },
    error_file: './logs/err.log',
    out_file: './logs/out.log',
    log_file: './logs/combined.log',
    time: true,
    max_memory_restart: '1G',
    node_args: '--max-old-space-size=2048',
    watch: false,
    ignore_watch: ['node_modules', 'logs', 'uploads', 'backups']
  }]
};
EOF
```

Step 4: Start Application with PM2

```
# Create logs directory
mkdir -p logs

# Start application
pm2 start ecosystem.config.js

# Save PM2 configuration
pm2 save

# Setup PM2 startup script
pm2 startup
# Follow the instructions provided by PM2
```

```
# Verify application is running
pm2 list
pm2 logs asmc-api
```

🗄️ Database Setup

Step 1: MongoDB Security Configuration

```
# Connect to MongoDB
mongo

# Create admin user
use admin
db.createUser({
  user: "admin",
  pwd: "secure_admin_password",
  roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
})

# Create application database and user
use asmc_prod
db.createUser({
  user: "asmc_user",
  pwd: "secure_asmc_password",
  roles: [
    { role: "readWrite", db: "asmc_prod" }
  ]
})

# Exit MongoDB
exit
```

Step 2: Configure MongoDB Security

```
# Edit MongoDB configuration
sudo nano /etc/mongod.conf
```

Security Configuration:

```
# /etc/mongod.conf
security:
    authorization: enabled

net:
    port: 27017
    bindIp: 127.0.0.1

storage:
    dbPath: /var/lib/mongodb
    journal:
```

```
        enabled: true

systemLog:
    destination: file
    logAppend: true
    path: /var/log/mongodb/mongod.log
```

```
# Restart MongoDB
sudo systemctl restart mongod

# Test connection with authentication
mongo -u asmc_user -p asmc_prod
```

Step 3: Create Database Indexes

```
# Connect to database
mongo -u asmc_user -p asmc_prod

# Create indexes for performance
db.members.createIndex({ "memberId": 1 }, { unique: true })
db.members.createIndex({ "personalInfo.email": 1 })
db.members.createIndex({ "personalInfo.phone": 1 })
db.members.createIndex({ "membership.status": 1 })

db.payments.createIndex({ "memberId": 1, "createdAt": -1 })
db.payments.createIndex({ "paymentStatus": 1 })
db.payments.createIndex({ "transactionId": 1 }, { unique: true })

db.hallBookings.createIndex({ "hallId": 1, "bookingDate": 1 })
db.hallBookings.createIndex({ "memberId": 1, "createdAt": -1 })

db.events.createIndex({ "eventDate": 1 })
db.events.createIndex({ "status": 1 })

# Exit MongoDB
exit
```

Nginx Configuration

Step 1: Create Nginx Configuration

```
# Create Nginx configuration
sudo nano /etc/nginx/sites-available/asmc-api
```

Nginx Configuration:

```
# /etc/nginx/sites-available/asmc-api
server {
    listen 80;
    server_name api.asmcdae.in;
```

```
    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "no-referrer-when-downgrade" always;
    add_header Content-Security-Policy "default-src 'self' http: https: data: blob:
'unsafe-inline'" always;

    # Rate limiting
    limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;
    limit_req zone=api burst=20 nodelay;

    # Proxy to Node.js application
    location / {
        proxy_pass http://localhost:7055;
        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;

        # Timeout settings
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;

        # Buffer settings
        proxy_buffering on;
        proxy_buffer_size 128k;
        proxy_buffers 4 256k;
        proxy_busy_buffers_size 256k;
    }

    # Static files
    location /public {
        alias /opt/asmc-api/public;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # Logs
    access_log /var/log/nginx/asmc-api.access.log;
    error_log /var/log/nginx/asmc-api.error.log;
}
```

Step 2: Enable Site

```
# Enable site
sudo ln -s /etc/nginx/sites-available/asmc-api /etc/nginx/sites-enabled/
```

```
# Remove default site
sudo rm /etc/nginx/sites-enabled/default

# Test configuration
sudo nginx -t

# Restart Nginx
sudo systemctl restart nginx
```

SSL Certificate Setup

Step 1: Install Certbot

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

# Get SSL certificate
sudo certbot --nginx -d api.asmcdae.in

# Test automatic renewal
sudo certbot renew --dry-run
```

Step 2: Configure Auto-Renewal

```
# Add cron job for auto-renewal
sudo crontab -e

# Add this line:
0 12 * * * /usr/bin/certbot renew --quiet
```

PM2 Configuration

Step 1: Advanced PM2 Configuration

```
# Create advanced ecosystem file
cat > ecosystem.config.js << 'EOF'
module.exports = {
  apps: [{
    name: 'asmc-api',
    script: 'app.js',
    instances: 'max',
    exec_mode: 'cluster',
    env: {
      NODE_ENV: 'production',
      PORT: 7055
    },
    error_file: './logs/err.log',
    out_file: './logs/out.log',
    log_file: './logs/combined.log',
```

```
    time: true,
    max_memory_restart: '1G',
    node_args: '--max-old-space-size=2048',
    watch: false,
    ignore_watch: ['node_modules', 'logs', 'uploads', 'backups'],
    min_uptime: '10s',
    max_restarts: 10,
    restart_delay: 4000,
    kill_timeout: 5000,
    wait_ready: true,
    listen_timeout: 10000,
    autorestart: true,
    env_production: {
      NODE_ENV: 'production',
      PORT: 7055
    }
  }]
};
EOF
```

Step 2: PM2 Monitoring Setup

```
# Install PM2 monitoring
pm2 install pm2-logrotate

# Configure log rotation
pm2 set pm2-logrotate:max_size 10M
pm2 set pm2-logrotate:retain 30
pm2 set pm2-logrotate:compress true

# Setup PM2 monitoring dashboard (optional)
pm2 install pm2-server-monit
```

Monitoring & Logging

Step 1: Application Logging

```
# Create log rotation configuration
sudo nano /etc/logrotate.d/asmc-api
```

Log Rotation Configuration:

```
/opt/asmc-api/logs/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 asmc asmc
    postrotate
```

```
        pm2 reloadLogs
    endscript
}
```

Step 2: System Monitoring

```
# Install system monitoring tools
sudo apt install -y htop iotop nethogs

# Monitor system resources
htop
```

Step 3: Health Check Script

```
# Create health check script
cat > health-check.sh << 'EOF'
#!/bin/bash

# Check if application is running
if ! pm2 list | grep -q "asmc-api.*online"; then
    echo "Application is not running"
    pm2 restart asmc-api
fi

# Check if MongoDB is running
if ! systemctl is-active --quiet mongod; then
    echo "MongoDB is not running"
    sudo systemctl start mongod
fi

# Check if Nginx is running
if ! systemctl is-active --quiet nginx; then
    echo "Nginx is not running"
    sudo systemctl start nginx
fi

echo "Health check completed"
EOF

chmod +x health-check.sh

# Add to crontab for regular health checks
crontab -e

# Add this line (every 5 minutes):
*/5 * * * * /opt/asmc-api/health-check.sh >> /opt/asmc-api/logs/health-check.log 2>&1
```

Backup Strategy

Step 1: Database Backup

```
# Create backup script
cat > backup-db.sh << 'EOF'
#!/bin/bash

BACKUP_DIR="/opt/asmc-api/backups"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="asmc_backup_$DATE"

# Create backup directory if it doesn't exist
mkdir -p $BACKUP_DIR

# Create database backup
mongodump --db asmc_prod --out $BACKUP_DIR/$BACKUP_FILE

# Compress backup
tar -czf $BACKUP_DIR/$BACKUP_FILE.tar.gz -C $BACKUP_DIR $BACKUP_FILE

# Remove uncompressed backup
rm -rf $BACKUP_DIR/$BACKUP_FILE

# Keep only last 30 days of backups
find $BACKUP_DIR -name "asmc_backup_*.tar.gz" -mtime +30 -delete

echo "Backup completed: $BACKUP_FILE.tar.gz"
EOF

chmod +x backup-db.sh

# Test backup
./backup-db.sh
```

Step 2: Automated Backup Schedule

```
# Add to crontab for daily backups at 2 AM
crontab -e

# Add this line:
0 2 * * * /opt/asmc-api/backup-db.sh >> /opt/asmc-api/logs/backup.log 2>&1
```

Docker Deployment

Step 1: Create Dockerfile

```
# Dockerfile
FROM node:18-alpine

# Set working directory
WORKDIR /app

# Copy package files
COPY package*.json ./
```

```
# Install dependencies
RUN npm ci --only=production

# Copy application code
COPY . .

# Create non-root user
RUN addgroup -g 1001 -S nodejs
RUN adduser -S nodejs -u 1001

# Create necessary directories
RUN mkdir -p logs uploads backups
RUN chown -R nodejs:nodejs /app

# Switch to non-root user
USER nodejs

# Expose port
EXPOSE 7055

# Health check
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
  CMD curl -f http://localhost:7055/health || exit 1

# Start application
CMD ["npm", "start"]
```

Step 2: Docker Compose

```
# docker-compose.yml
version: '3.8'

services:
    asmc-api:
        build: .
        ports:
            - '7055:7055'
        environment:
            - NODE_ENV=production
            - MONGO_URI=mongodb://mongo:27017/asmc
        depends_on:
            - mongo
        volumes:
            - ./uploads:/app/uploads
            - ./backups:/app/backups
            - ./logs:/app/logs
        restart: unless-stopped

    mongo:
        image: mongo:6.0
        ports:
```

```
            - '27017:27017'
        volumes:
            - mongo_data:/data/db
        environment:
            - MONGO_INITDB_ROOT_USERNAME=admin
            - MONGO_INITDB_ROOT_PASSWORD=password
        restart: unless-stopped

    nginx:
        image: nginx:alpine
        ports:
            - '80:80'
            - '443:443'
        volumes:
            - ./nginx.conf:/etc/nginx/nginx.conf
            - ./ssl:/etc/nginx/ssl
        depends_on:
            - asmc-api
        restart: unless-stopped

volumes:
    mongo_data:
```

Step 3: Deploy with Docker

```
# Build and start services
docker-compose up -d

# View logs
docker-compose logs -f asmc-api

# Stop services
docker-compose down
```

Troubleshooting

Common Deployment Issues

1. Application Won't Start

```
# Check PM2 logs
pm2 logs asmc-api

# Check application logs
tail -f /opt/asmc-api/logs/combined.log

# Restart application
pm2 restart asmc-api
```

2. Database Connection Issues

```
# Check MongoDB status
sudo systemctl status mongod

# Check MongoDB logs
sudo tail -f /var/log/mongodb/mongod.log

# Test connection
mongo -u asmc_user -p asmc_prod
```

3. Nginx Issues

```
# Check Nginx configuration
sudo nginx -t

# Check Nginx logs
sudo tail -f /var/log/nginx/error.log

# Restart Nginx
sudo systemctl restart nginx
```

4. SSL Certificate Issues

```
# Check certificate status
sudo certbot certificates

# Renew certificate
sudo certbot renew

# Check certificate expiry
openssl x509 -in /etc/letsencrypt/live/api.asmcdae.in/cert.pem -text -noout | grep
"Not After"
```

Performance Optimization

1. Database Optimization

```
# Monitor database performance
mongo --eval "db.stats()"

# Check slow queries
mongo --eval "db.setProfilingLevel(2, { slowms: 100 })"
```

2. Application Optimization

```
# Monitor PM2 processes
pm2 monit

# Check memory usage
free -h
df -h
```

3. Nginx Optimization

```
# Check Nginx status
sudo systemctl status nginx

# Monitor connections
sudo netstat -tulpn | grep :80
sudo netstat -tulpn | grep :443
```

Next Steps

After successful deployment:

1. **Set up monitoring**: Implement comprehensive monitoring and alerting
2. **Configure backups**: Set up automated database and file backups
3. **Security hardening**: Implement additional security measures
4. **Performance tuning**: Optimize for your specific load patterns
5. **Documentation**: Update documentation with your specific configuration

Support

If you encounter deployment issues:

1. **Check logs**: Review application, system, and database logs
2. **Verify configuration**: Ensure all environment variables and configurations are correct
3. **Test connectivity**: Verify network connectivity and firewall settings
4. **Resource monitoring**: Check server resources (CPU, memory, disk space)

Congratulations! Your ASMC API is now deployed and running in production!