

ASMC Admin Panel - Deployment Guide

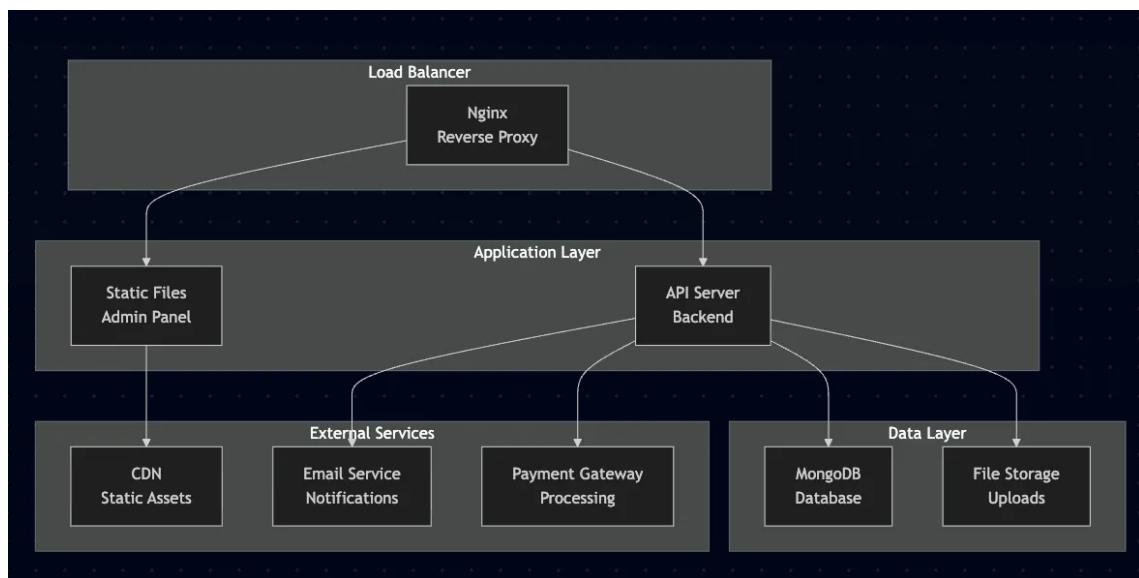
This document provides comprehensive instructions for deploying the ASMC Admin Panel in various environments, including development, staging, and production.

Table of Contents

- [Deployment Overview](#)
- [Environment Preparation](#)
- [Build Process](#)
- [Static File Deployment](#)
- [Nginx Configuration](#)
- [Docker Deployment](#)
- [CI/CD Pipeline](#)
- [SSL Configuration](#)
- [Monitoring & Maintenance](#)
- [Troubleshooting](#)

Deployment Overview

Deployment Architecture



Deployment Options

1. **Static File Hosting:** Direct file serving
2. **Nginx Web Server:** Production-grade serving
3. **Docker Container:** Containerized deployment
4. **Cloud Platforms:** AWS, Azure, GCP
5. **CDN Integration:** Global content delivery

Environment Preparation

Server Requirements

Minimum Requirements

- **OS:** Ubuntu 20.04 LTS or CentOS 8+
- **RAM:** 2GB minimum, 4GB recommended
- **Storage:** 20GB SSD
- **CPU:** 2 cores
- **Network:** 100 Mbps

Recommended Requirements

- **OS:** Ubuntu 22.04 LTS
- **RAM:** 8GB
- **Storage:** 50GB SSD
- **CPU:** 4 cores
- **Network:** 1 Gbps

Software Dependencies

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

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

# Install Nginx
sudo apt install nginx -y

# Install PM2 (for backend)
sudo npm install -g pm2

# Install Git
sudo apt install git -y

# Install Certbot (for SSL)
sudo apt install certbot python3-certbot-nginx -y
```

Directory Structure

```
# Create application directories
sudo mkdir -p /var/www/asmc-admin
sudo mkdir -p /var/log/asmc-admin
sudo mkdir -p /etc/nginx/sites-available
sudo mkdir -p /etc/nginx/sites-enabled

# Set permissions
sudo chown -R www-data:www-data /var/www/asmc-admin
sudo chmod -R 755 /var/www/asmc-admin
```

Build Process

Local Build

```
# Navigate to project directory
cd /path/to/asmc-admin

# Install dependencies
npm ci --only=production

# Create production build
npm run build

# Verify build
ls -la build/
```

Environment-Specific Builds

Development Build

```
# Development environment
NODE_ENV=development npm run build
```

Production Build

```
# Production environment with optimizations
NODE_ENV=production npm run build

# Build with bundle analysis
npm run build && npx webpack-bundle-analyzer build/static/js/*.js
```

Build Optimization

```
# Analyze bundle size
npm install -g webpack-bundle-analyzer
npm run build
npx webpack-bundle-analyzer build/static/js/*.js

# Check for unused dependencies
npm install -g depcheck
depcheck

# Optimize images
npm install -g imagemin-cli
imagemin src/assets/images/* --out-dir=build/static/images
```

Static File Deployment

Direct File Copy

```
# Copy build files to web server
sudo cp -r build/* /var/www/asmc-admin/

# Set proper permissions
sudo chown -R www-data:www-data /var/www/asmc-admin
```

```

sudo chmod -R 755 /var/www/asmc-admin

# Create backup
sudo cp -r /var/www/asmc-admin /var/www/asmc-admin-backup-$(date +%Y%m%d)

```

Automated Deployment Script

```

#!/bin/bash
# deploy.sh

set -e

echo "Starting deployment..."

# Variables
APP_DIR="/var/www/asmc-admin"
BACKUP_DIR="/var/www/asmc-admin-backup-$(date +%Y%m%d-%H%M%S)"
BUILD_DIR="build"

# Create backup
if [ -d "$APP_DIR" ]; then
    echo "Creating backup..."
    sudo cp -r "$APP_DIR" "$BACKUP_DIR"
fi

# Build the application
echo "Building application..."
npm ci --only=production
npm run build

# Deploy to server
echo "Deploying to server..."
sudo rm -rf "$APP_DIR"
sudo cp -r "$BUILD_DIR" "$APP_DIR"

# Set permissions
sudo chown -R www-data:www-data "$APP_DIR"
sudo chmod -R 755 "$APP_DIR"

# Restart services
echo "Restarting Nginx..."
sudo systemctl reload nginx

echo "Deployment completed successfully!"

```

Nginx Configuration

Basic Configuration

Location: /etc/nginx/sites-available/asmc-admin

```

server {
    listen 80;
    server_name admin.asmc.com;
    root /var/www/asmc-admin;
    index index.html;

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;

    # Handle client-side routing
    location / {
        try_files $uri $uri/ /index.html;
    }

    # Cache static assets
    location /static/ {
        expires 1y;
        add_header Cache-Control "public, immutable";
        access_log off;
    }

    # Cache images
    location ~* \.(jpg|jpeg|png|gif|ico|svg)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
        access_log off;
    }

    # Cache CSS and JS
    location ~* \.(css|js)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
        access_log off;
    }

    # API proxy to backend
    location /api {
        proxy_pass http://localhost:7055;
        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;

        # WebSocket support
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}

```

```

# Gzip compression
gzip on;
gzip_vary on;
gzip_min_length 1024;
gzip_types
    text/plain
    text/css
    text/xml
    text/javascript
    application/javascript
    application/xml+rss
    application/json;

# Error pages
error_page 404 /index.html;
error_page 500 502 503 504 /50x.html;
location = /50x.html {
    root /usr/share/nginx/html;
}
}

```

Advanced Configuration

```

# Rate limiting
limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;
limit_req_zone $binary_remote_addr zone=login:10m rate=5r/m;

server {
    listen 80;
    server_name admin.asmc.com;

    # Rate limiting
    limit_req zone=api burst=20 nodelay;

    location /api/auth/login {
        limit_req zone=login burst=5 nodelay;
        proxy_pass http://localhost:7055;
    }

    # Security headers
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
    add_header Content-Security-Policy "default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data: https:; font-src 'self' data:;" always;

    # CORS headers for API
    location /api {
        add_header Access-Control-Allow-Origin "https://admin.asmc.com" always;
        add_header Access-Control-Allow-Methods "GET, POST, PUT, DELETE, OPTIONS"
always;
        add_header Access-Control-Allow-Headers "Authorization, Content-Type" always;
        add_header Access-Control-Allow-Credentials true always;
    }
}

```

```

if ($request_method = 'OPTIONS') {
    return 204;
}

proxy_pass http://localhost:7055;
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;
}

# Logging
access_log /var/log/nginx/asmc-admin.access.log;
error_log /var/log/nginx/asmc-admin.error.log;
}

```

Enable Site

```

# Create symlink
sudo ln -s /etc/nginx/sites-available/asmc-admin /etc/nginx/sites-enabled/

# Test configuration
sudo nginx -t

# Reload Nginx
sudo systemctl reload nginx

# Enable auto-start
sudo systemctl enable nginx

```

Docker Deployment

Dockerfile

```

# Multi-stage build
FROM node:18-alpine as build

# Set working directory
WORKDIR /app

# Copy package files
COPY package*.json ./

# Install dependencies
RUN npm ci --only=production

# Copy source code
COPY . .

# Build the application

```

```

RUN npm run build

# Production stage
FROM nginx:alpine

# Copy built files
COPY --from=build /app/build /usr/share/nginx/html

# Copy nginx configuration
COPY nginx.conf /etc/nginx/nginx.conf

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

# Set permissions
RUN chown -R nextjs:nodejs /usr/share/nginx/html

# Expose port
EXPOSE 80

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

# Start nginx
CMD ["nginx", "-g", "daemon off;"]

```

nginx.conf for Docker

```

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    # Logging
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;
    error_log /var/log/nginx/error.log;

    # Gzip compression
    gzip on;
    gzip_vary on;
    gzip_min_length 1024;
    gzip_types text/plain text/css application/json application/javascript text/xml
application/xml application/xml+rss text/javascript;

```

```

server {
    listen 80;
    server_name localhost;
    root /usr/share/nginx/html;
    index index.html;

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;

    # Handle client-side routing
    location / {
        try_files $uri $uri/ /index.html;
    }

    # Cache static assets
    location /static/ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # API proxy
    location /api {
        proxy_pass http://asmc-api:7055;
        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;
    }
}
}

```

Docker Compose

```

version: '3.8'

services:
  asmc-admin:
    build: .
    ports:
      - '3000:80'
    environment:
      - REACT_APP_API_BASE_URL=http://asmc-api:7055/api
    depends_on:
      - asmc-api
    restart: unless-stopped
    networks:
      - asmc-network

  asmc-api:

```

```

image: asmc-api:latest
ports:
  - '7055:7055'
environment:
  - NODE_ENV=production
  - MONGODB_URI=mongodb://mongodb:27017/asmc
depends_on:
  - mongodb
restart: unless-stopped
networks:
  - asmc-network

mongodb:
  image: mongo:6.0
  ports:
    - '27017:27017'
  environment:
    - MONGO_INITDB_ROOT_USERNAME=admin
    - MONGO_INITDB_ROOT_PASSWORD=password
  volumes:
    - mongodb_data:/data/db
  restart: unless-stopped
  networks:
    - asmc-network

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

volumes:
  mongodb_data:

networks:
  asmc-network:
    driver: bridge

```

Docker Deployment Commands

```

# Build and run with Docker Compose
docker-compose up -d

```

```

# Build specific service
docker-compose build asmc-admin

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

# Scale services
docker-compose up -d --scale asmc-admin=3

# Update services
docker-compose pull
docker-compose up -d

# Stop services
docker-compose down

```

CI/CD Pipeline

GitHub Actions

.github/workflows/deploy.yml

```

name: Deploy ASMC Admin

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3

        - name: Setup Node.js
          uses: actions/setup-node@v3
          with:
            node-version: '18'
            cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run tests
        run: npm test

      - name: Run linting
        run: npm run lint

```

```

build:
  needs: test
  runs-on: ubuntu-latest

  steps:
    - uses: actions/checkout@v3

    - name: Setup Node.js
      uses: actions/setup-node@v3
      with:
        node-version: '18'
        cache: 'npm'

    - name: Install dependencies
      run: npm ci

    - name: Build application
      run: npm run build
      env:
        REACT_APP_API_BASE_URL: ${{ secrets.API_BASE_URL }}
        REACT_APP_ENVIRONMENT: production

    - name: Upload build artifacts
      uses: actions/upload-artifact@v3
      with:
        name: build-files
        path: build/

deploy:
  needs: build
  runs-on: ubuntu-latest
  if: github.ref == 'refs/heads/main'

  steps:
    - name: Download build artifacts
      uses: actions/download-artifact@v3
      with:
        name: build-files
        path: build/

    - name: Deploy to server
      uses: appleboy/ssh-action@v0.1.5
      with:
        host: ${{ secrets.HOST }}
        username: ${{ secrets.USERNAME }}
        key: ${{ secrets.SSH_KEY }}
        script: |
          # Create backup
          sudo cp -r /var/www/asmc-admin /var/www/asmc-admin-backup-$(date
          +%Y%m%d-%H%M%S)

          # Deploy new files

```

```

        sudo rm -rf /var/www/asmc-admin/*
        sudo cp -r build/* /var/www/asmc-admin/

        # Set permissions
        sudo chown -R www-data:www-data /var/www/asmc-admin
        sudo chmod -R 755 /var/www/asmc-admin

        # Reload Nginx
        sudo systemctl reload nginx

```

Jenkins Pipeline

Jenkinsfile

```

pipeline {
    agent any

    environment {
        NODE_VERSION = '18'
        BUILD_DIR = 'build'
        DEPLOY_DIR = '/var/www/asmc-admin'
    }

    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }

        stage('Install Dependencies') {
            steps {
                sh 'npm ci'
            }
        }

        stage('Run Tests') {
            steps {
                sh 'npm test -- --coverage --watchAll=false'
            }
            post {
                always {
                    publishTestResults testResultsPattern: 'test-results.xml'
                    publishCoverage adapters: [
                        jacocoAdapter('coverage/lcov.info')
                    ]
                }
            }
        }

        stage('Lint') {
            steps {
                sh 'npm run lint'
            }
        }
    }
}

```

```

        }

    }

    stage('Build') {
        steps {
            sh 'npm run build'
        }
        post {
            success {
                archiveArtifacts artifacts: 'build/**/*', fingerprint: true
            }
        }
    }

    stage('Deploy to Staging') {
        when {
            branch 'develop'
        }
        steps {
            sh '''
                ssh staging-server "sudo cp -r ${DEPLOY_DIR} ${DEPLOY_DIR}-
                backup-$(date +%Y%m%d-%H%M%S)"
                rsync -avz --delete build/ staging-server:${DEPLOY_DIR}/
                ssh staging-server "sudo chown -R www-data:www-data ${DEPLOY_DIR}
                && sudo systemctl reload nginx"
            '''
        }
    }

    stage('Deploy to Production') {
        when {
            branch 'main'
        }
        steps {
            sh '''
                ssh production-server "sudo cp -r ${DEPLOY_DIR} ${DEPLOY_DIR}-
                backup-$(date +%Y%m%d-%H%M%S)"
                rsync -avz --delete build/ production-server:${DEPLOY_DIR}/
                ssh production-server "sudo chown -R www-data:www-data
                ${DEPLOY_DIR} && sudo systemctl reload nginx"
            '''
        }
    }

    post {
        always {
            cleanWs()
        }
        failure {
            emailext (
                subject: "Build Failed: ${env.JOB_NAME} - ${env.BUILD_NUMBER}",

```

```

        body: "Build failed. Check console output at ${env.BUILD_URL}",
        to: "${env.CHANGE_AUTHOR_EMAIL}"
    )
}
}
}

```

SSL Configuration

Let's Encrypt SSL

```

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

# Get SSL certificate
sudo certbot --nginx -d admin.asmc.com

# Test renewal
sudo certbot renew --dry-run

# Auto-renewal cron job
sudo crontab -e
# Add: 0 12 * * * /usr/bin/certbot renew --quiet

```

SSL Configuration with Nginx

```

server {
    listen 80;
    server_name admin.asmc.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name admin.asmc.com;

    # SSL configuration
    ssl_certificate /etc/letsencrypt/live/admin.asmc.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/admin.asmc.com/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/admin.asmc.com/chain.pem;

    # SSL security
    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;

    # HSTS
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
}

```

```
# Rest of configuration...
}
```

Monitoring & Maintenance

Health Checks

```
#!/bin/bash
# health-check.sh

# Check if Nginx is running
if ! systemctl is-active --quiet nginx; then
    echo "Nginx is not running"
    exit 1
fi

# Check if application files exist
if [ ! -f "/var/www/asmc-admin/index.html" ]; then
    echo "Application files not found"
    exit 1
fi

# Check if API is responding
if ! curl -f -s http://localhost/api/health > /dev/null; then
    echo "API is not responding"
    exit 1
fi

echo "Health check passed"
exit 0
```

Log Monitoring

```
# Create log monitoring script
#!/bin/bash
# monitor-logs.sh

LOG_FILE="/var/log/nginx/asmc-admin.error.log"
ALERT_THRESHOLD=10

# Check for errors in the last 5 minutes
ERROR_COUNT=$(tail -n 1000 "$LOG_FILE" | grep "$(date -d '5 minutes ago' '+%Y/%m/%d %H:%M')" | grep -c "error")

if [ "$ERROR_COUNT" -gt "$ALERT_THRESHOLD" ]; then
    echo "High error rate detected: $ERROR_COUNT errors in the last 5 minutes"
    # Send alert email or notification
    mail -s "High Error Rate Alert" admin@asmc.com << EOF
High error rate detected on ASMC Admin Panel:
- Errors: $ERROR_COUNT
EOF
```

```
- Time: $(date)
- Threshold: $ALERT_THRESHOLD
EOF
fi
```

Automated Backups

```
#!/bin/bash
# backup.sh

BACKUP_DIR="/backups/asmc-admin"
APP_DIR="/var/www/asmc-admin"
DATE=$(date +%Y%m%d-%H%M%S)

# Create backup directory
mkdir -p "$BACKUP_DIR"

# Create backup
tar -czf "$BACKUP_DIR/asmc-admin-$DATE.tar.gz" -C "$APP_DIR" .

# Keep only last 7 days of backups
find "$BACKUP_DIR" -name "asmc-admin-*tar.gz" -mtime +7 -delete

echo "Backup created: asmc-admin-$DATE.tar.gz"
```

Troubleshooting

Common Issues

1. 404 Errors on Refresh

Problem: Page shows 404 when refreshed

Solution:

```
# Ensure Nginx configuration includes:
location / {
    try_files $uri $uri/ /index.html;
}
```

2. API Connection Issues

Problem: Cannot connect to backend API

Solution:

```
# Check if backend is running
curl http://localhost:7055/api/health

# Check Nginx proxy configuration
sudo nginx -t

# Check firewall
sudo ufw status
```

3. Permission Issues

Problem: Files not accessible

Solution:

```
# Fix permissions
sudo chown -R www-data:www-data /var/www/asmc-admin
sudo chmod -R 755 /var/www/asmc-admin

# Check SELinux (if applicable)
sudo setsebool -P httpd_can_network_connect 1
```

4. SSL Certificate Issues

Problem: SSL certificate not working

Solution:

```
# Check certificate status
sudo certbot certificates

# Renew certificate
sudo certbot renew

# Test SSL
openssl s_client -connect admin.asmc.com:443
```

Performance Issues

1. Slow Loading Times

Solution:

```
# Enable gzip compression
# Add to Nginx config:
gzip on;
gzip_types text/css application/javascript application/json;

# Check file sizes
du -sh /var/www/asmc-admin/*

# Optimize images
npm install -g imagemin-cli
imagemin src/assets/images/* --out-dir=build/static/images
```

2. High Memory Usage

Solution:

```
# Check memory usage
free -h
ps aux --sort=-%mem | head

# Optimize Nginx worker processes
```

```
# In /etc/nginx/nginx.conf:  
worker_processes auto;  
worker_connections 1024;
```

Debug Commands

```
# Check Nginx status  
sudo systemctl status nginx  
sudo nginx -t  
  
# View logs  
sudo tail -f /var/log/nginx/asmc-admin.error.log  
sudo tail -f /var/log/nginx/asmc-admin.access.log  
  
# Check disk space  
df -h  
  
# Check network connectivity  
ping admin.asmc.com  
curl -I https://admin.asmc.com  
  
# Check SSL certificate  
openssl s_client -connect admin.asmc.com:443 -servername admin.asmc.com
```

Summary

The ASMC Admin Panel deployment guide covers:

- **Multiple Deployment Options:** Static files, Nginx, Docker
- **Production Configuration:** SSL, security, performance
- **CI/CD Integration:** Automated testing and deployment
- **Monitoring:** Health checks, logging, backups
- **Troubleshooting:** Common issues and solutions

This comprehensive guide ensures reliable, secure, and maintainable deployment of the admin panel.

Version: 1.0.0

Last Updated: January 2025

Maintainer: ASMC Development Team