# Docker Publishing Guide for GLSP Generator

This guide covers how to build, test, and publish the GLSP Generator as a Docker container.

## Prerequisites

* Docker installed and running
* Docker Hub account (or other registry)
* Docker Buildx for multi-architecture builds
* Repository access for publishing

## Quick Start

### Local Development

# Build the image locally  
yarn docker:build  
  
# Test the image  
yarn docker:test  
  
# Run CLI mode  
docker run --rm -v $(pwd):/workspace glsp-generator generate /workspace/grammar.langium  
  
# Run API mode  
yarn docker:run:api

### Publishing

# Build, tag, and push (requires Docker Hub login)  
yarn docker:publish  
  
# Or step by step:  
yarn docker:build  
yarn docker:tag  
yarn docker:push

## Detailed Publishing Process

### 1. Setup Docker Hub Account

1. Create account at <https://hub.docker.com>
2. Create repository: johnholliday/glsp-generator
3. Generate access token: Account Settings → Security → New Access Token
4. Login locally:

* docker login  
  # Username: johnholliday  
  # Password: <your-access-token>

### 2. Build Multi-Architecture Image

# Setup buildx (one-time)  
docker buildx create --name mybuilder --use  
docker buildx inspect --bootstrap  
  
# Build for multiple platforms  
yarn docker:build:multiarch  
  
# Or manually:  
docker buildx build \  
 --platform linux/amd64,linux/arm64 \  
 -t johnholliday/glsp-generator:latest \  
 --push .

### 3. Version Tagging Strategy

The docker-tag.js script automatically creates these tags:

* latest - Always points to newest version
* 2.1.171 - Exact version from package.json
* 2.1 - Minor version (receives patch updates)
* 2 - Major version (receives minor/patch updates)

# Tag images based on package.json version  
yarn docker:tag  
  
# Manual tagging:  
docker tag glsp-generator:latest johnholliday/glsp-generator:2.1.171  
docker tag glsp-generator:latest johnholliday/glsp-generator:2.1  
docker tag glsp-generator:latest johnholliday/glsp-generator:2  
docker tag glsp-generator:latest johnholliday/glsp-generator:latest

### 4. Push to Registry

# Push all tags  
yarn docker:push  
  
# Or push individually:  
docker push johnholliday/glsp-generator:2.1.171  
docker push johnholliday/glsp-generator:2.1  
docker push johnholliday/glsp-generator:2  
docker push johnholliday/glsp-generator:latest

### 5. GitHub Container Registry (Alternative)

# Login to ghcr.io  
echo $GITHUB\_TOKEN | docker login ghcr.io -u USERNAME --password-stdin  
  
# Tag for GHCR  
docker tag glsp-generator:latest ghcr.io/johnholliday/glsp-generator:latest  
  
# Push to GHCR  
docker push ghcr.io/johnholliday/glsp-generator:latest

## Automated Publishing with GitHub Actions

The repository includes .github/workflows/docker-publish.yml that automatically:

1. **On Push to Main**: Builds and publishes with latest tag
2. **On Version Tag**: Publishes with version tags (e.g., v2.1.171)
3. **On Pull Request**: Builds but doesn't push (for testing)

### Required GitHub Secrets

Set these in repository Settings → Secrets:

* DOCKER\_USERNAME - Docker Hub username
* DOCKER\_TOKEN - Docker Hub access token
* GITHUB\_TOKEN - Automatically provided by GitHub Actions

### Triggering Automated Publish

# Tag a new version  
git tag v2.1.171  
git push origin v2.1.171  
  
# Or create a release on GitHub  
# This triggers the workflow automatically

## Build Arguments and Labels

The Dockerfile accepts these build arguments:

ARG VERSION=latest  
ARG BUILD\_DATE  
ARG VCS\_REF

Build with arguments:

docker build \  
 --build-arg VERSION=$(node -p "require('./package.json').version") \  
 --build-arg BUILD\_DATE=$(date -u +'%Y-%m-%dT%H:%M:%SZ') \  
 --build-arg VCS\_REF=$(git rev-parse HEAD) \  
 -t glsp-generator .

## Testing Published Images

### Basic Functionality

# Test CLI  
docker run --rm johnholliday/glsp-generator:latest --version  
docker run --rm johnholliday/glsp-generator:latest --help  
  
# Test generation  
docker run --rm -v $(pwd):/workspace \  
 johnholliday/glsp-generator:latest \  
 generate /workspace/test.langium

### API Mode

# Run API server  
docker run -d -p 3000:3000 --name glsp-api \  
 johnholliday/glsp-generator:latest \  
 node dist/api-server.js  
  
# Test health endpoint  
curl http://localhost:3000/health  
  
# Stop and remove  
docker stop glsp-api && docker rm glsp-api

### Security Scanning

# Scan with Trivy  
trivy image johnholliday/glsp-generator:latest  
  
# Scan with Docker Scout  
docker scout cves johnholliday/glsp-generator:latest

## Troubleshooting

### Build Failures

# Clear builder cache  
docker buildx prune -f  
  
# Build with no cache  
docker build --no-cache -t glsp-generator .  
  
# Check build logs  
docker buildx build --progress=plain .

### Push Failures

# Check login status  
docker login  
  
# Check rate limits  
TOKEN=$(curl -s "https://auth.docker.io/token?service=registry.docker.io&scope=repository:ratelimitpreview/test:pull" | jq -r .token)  
curl -s -H "Authorization: Bearer $TOKEN" https://registry-1.docker.io/v2/ratelimitpreview/test/manifests/latest | jq  
  
# Retry with specific tag  
docker push johnholliday/glsp-generator:latest --disable-content-trust

### Multi-arch Issues

# List current builders  
docker buildx ls  
  
# Create new builder  
docker buildx create --name newbuilder --driver docker-container --use  
  
# Inspect platforms  
docker buildx imagetools inspect johnholliday/glsp-generator:latest

## Best Practices

1. **Version Bumping**: Always update package.json version before publishing
2. **Testing**: Run yarn docker:test before publishing
3. **Security**: Scan images before publishing
4. **Documentation**: Update DOCKER.md if usage changes
5. **Tagging**: Follow semantic versioning
6. **Multi-arch**: Always build for amd64 and arm64

## Publishing Checklist

* Update version in package.json
* Build and test locally
* Run security scan
* Login to Docker Hub
* Run yarn docker:publish
* Test published image
* Update documentation if needed
* Create GitHub release

## Useful Commands

# View image details  
docker inspect johnholliday/glsp-generator:latest  
  
# View image history  
docker history johnholliday/glsp-generator:latest  
  
# Export image  
docker save johnholliday/glsp-generator:latest | gzip > glsp-generator.tar.gz  
  
# Import image  
docker load < glsp-generator.tar.gz  
  
# Remove all local images  
docker rmi $(docker images -q johnholliday/glsp-generator)