Docker Deployment Guide

Run the FrameworX runtime inside Docker for container-based deployments.

Reference Runtime Installation Deployment Docker

The runtime ships on .NET 10 and runs as a long-lived container process. This page covers the image Tatsoft publishes on Docker Hub, preparing an image you build yourself, the runtime configuration needed inside a container, and persistent-volume layout.

Published Image
Runtime Requirements
Dockerfile Example
Kubernetes Manifest
Persistent Volumes
Cloud Marketplace Images
Troubleshooting

Published Image

Tatsoft publishes an image on Docker Hub, `tatsoftdockerhub/frameworx`, tagged by version, with `latest` for the current release. It runs the Solution Center service on port **10108** and starts with no solution loaded. You upload, start and license solutions from Solution Center on a Windows machine that connects to the container.

Run the Container

```
docker pull tatsoftdockerhub/frameworx:latest
docker run -d --name frameworx -p 10108:10108 -p 3101:3101 -v frameworx-solutions:/Documents/FrameworX
/Solutions tatsoftdockerhub/frameworx:latest
```

On Linux with Docker Engine, prefix each command with `sudo` unless your user is in the `docker` group. Docker Desktop on Windows and macOS runs them as written.

Solution Volume

Mount the volume on `/Documents/FrameworX/Solutions`. The Solution Center service keeps each solution there, together with its alarm, historian, retentive and runtime-user databases. The image declares no volume. Without this mount, or with the volume mounted on any other path, those files stay inside the container and are deleted when the container is removed.

Ports

Publish port 10108 for the Solution Center service, and the port each solution runs on. A solution runs on the Main Port set in **Runtime Startup**. Set Main Port to the published port, 3101 in the command above, and clear **Auto Ports** so the runtime keeps that port. Ports 3201 and 3301 are needed only for solutions that run on them.

Deploy a Solution

1. In the Designer on the Windows machine, set the solution to Multiplatform at **Solution Settings Target Platform and Product Modify**. A solution with a Windows target does not run in the container. Then run **Runtime Build and Publish**.
2. In Solution Center, go to **Server Information Solutions Location Solution Server**, enter `http://<host>:10108`, leave Access Key empty, and click **Connect**. Connected Server Information then shows the server platform, for example Linux-x64.
3. Use **Server Management Upload File** to send the `.dbsln` file.
4. Open the uploaded solution, set Main Port and Auto Ports as described above, and click **Run Startup**. The command is sent to the container, and the status in Solution Center updates once the container finishes starting the solution.

License

While Solution Center is connected to the container, **Your License Online Activation Activate** licenses the container, not the Windows machine. An activation code activates one installation at a time. To move it to the container, first deactivate it on the installation that holds it.

After a Container Restart

Restarting the container brings back the Solution Center service, not the solution. Start the solution again from Solution Center with **Run Startup**.

Runtime Requirements

- .NET 10 runtime. The runtime targets `net10.0`.
- Linux or Windows container host. Both official .NET 10 base images work.

- Writable volume for the solution folder, log folder, and Historian storage.
- Exposed TCP port for the runtime HTTP endpoint. The conventional port for the runtime service is **3101**. Each service JSON sets its own `portNumber`.
- Network reachability to any connected PLCs, databases, MQTT brokers, and tRPC clients.

Image Layout

For an image you build yourself, place the runtime binaries under the container path `/app/FactoryStudio` and the solution under a mounted volume, for example `/solutions/<SolutionName>`. Log files go to `/app/FactoryStudio/Logs`.

Dockerfile Example

```
FROM mcr.microsoft.com/dotnet/runtime:10.0

WORKDIR /app

COPY ./FactoryStudio /app/FactoryStudio
COPY ./Solution /solutions/Plant1

EXPOSE 3101

ENTRYPOINT ["dotnet", "/app/FactoryStudio/bin/net10.0/TStartup.dll", "/solution:/solutions/Plant1.tproj"]
```

Adjust the ENTRYPOINT to match the runtime service bundled in your image.

Kubernetes Manifest

A minimal deployment manifest. The runtime exposes `/health` and `/ready` on the listener port. Wire them as Kubernetes `livenessProbe` and `readinessProbe`. Both endpoints return JSON with the product version, uptime, and process start timestamp; full payload contract on the FrameworX Enterprise IT Readiness Reference.

The manifest runs an image built as in the Dockerfile example, which starts the runtime directly. The published image starts the Solution Center service instead, with no solution loaded, so it answers nothing on port 3101 until a solution is started.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frameworx-runtime
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: runtime
        image: <your-registry>/frameworx-runtime:<tag>
        ports:
        - containerPort: 3101
        livenessProbe:
          httpGet: { path: /health, port: 3101 }
          initialDelaySeconds: 10
          periodSeconds: 30
        readinessProbe:
          httpGet: { path: /ready, port: 3101 }
          initialDelaySeconds: 5
          periodSeconds: 10
```

Persistent Volumes

For an image you build yourself, as in the Dockerfile example. For the published image, see Solution Volume above.

Path	Purpose	Mount Type
`/solutions`	Solution files (`.tproj`) and embedded datasets.	Persistent volume.
`/app/FactoryStudio/Logs`	Runtime trace logs.	Persistent volume or a log-forwarding sidecar.
`/app/FactoryStudio/Historian`	Built-in Historian storage when the solution uses it.	Persistent volume with IOPS matching your retention load.
`/app/FactoryStudio/certs`	PFX files for HTTPS. See TLS and SSL Configuration.	Secret-backed volume.

Cloud Marketplace Images

Tatsoft plans to publish prebuilt runtime images for Azure, AWS, and GCP marketplaces. Images will ship with environment variables for the solution path, HTTP port, and license activation, a default non-root user, and a minimal base image. Point your orchestrator at the marketplace image tag for your target runtime version, mount a solution volume, and expose the HTTP port. No repackaging is required.

Troubleshooting

Symptom	Likely Cause	Next Step
Container exits immediately.	Solution path wrong, license missing, or entrypoint pointing at the wrong DLL.	Check `docker logs` for the startup trace, and verify the solution volume mount.
Runtime cannot reach PLCs or databases.	Network policy or missing service in the pod network.	Confirm the pod network reaches the target, and check outbound firewall rules.
Historian writes fail.	Persistent volume read-only, or not mounted.	Verify the Historian volume mount and permissions on `/app/FactoryStudio/Historian`.
The solution and its history are gone after the container was recreated.	The volume was not mounted on `/Documents/FrameworX/Solutions`.	Mount it on that path. See Solution Volume above.
The web client or the REST API does not answer on the published port.	The solution's Main Port differs from the published port.	Set Main Port to the published port and clear Auto Ports in Runtime Startup, then run the solution again.

In this section...