Redundancy in Docker Bridge Networks

Configure a redundant pair when the runtime runs inside Docker containers, and understand why a bridge-network pair silently starts as two independent nodes.

Reference **Deployment** Redundancy Docker Bridge Networks

Version 10.1.5+

A redundant pair is formed at startup, from the redundancy addresses configured in the solution. Inside a container the runtime sees only the container network identity, not the host network identity, so the addresses that work on bare metal do not automatically work in a container. This page states the supported configuration for each Docker network mode.

The Rule That Governs This
Symptom: Both Nodes Report Active
Option 1: Host Networking
Option 2: Bridge Networking Across Two Hosts
Verification Checklist
Related Pages

The Rule That Governs This

When the platform launches the runtime, it passes the redundancy startup parameters `/redundancy`, `/ip1` and `/ip2` **only when at least one of the two configured redundancy addresses is local to the machine that is launching the runtime**. If neither address is local, the platform falls back to the single-host startup form, with `/port1` only.

This is deliberate. A node that is neither IP1 nor IP2 cannot hold a role in the pair, so the runtime starts as a standalone node instead of joining a pair it cannot be a member of.

Inside a Docker container, "local" means local to the container, not to the Docker host.

Network mode	Addresses local to the runtime	Effect on a pair configured with the two host LAN addresses
`bridge` (default, and what Docker Compose creates)	Only the container internal address, for example `172.17.0.2` or `172.20.0.5`.	Neither configured address is local. Redundancy parameters are withheld and each node starts standalone.
`host`	The Docker host addresses, because the container shares the host network stack.	Works unchanged. The ordinary bare-metal configuration applies.
`macvlan` / overlay with a routable container address	The address assigned to the container on that network.	Works when the configured redundancy addresses are the container addresses.

Symptom: Both Nodes Report Active

> **The degradation is not raised in the interface**
>
> A pair configured with two addresses that are not local to their containers does not fail loudly. Each node starts, runs normally, and reports itself as the active node. There is no error, no alarm and no warning in the interface, so the visible sign is that there are two active nodes and no synchronization between them.
>
> From *FrameworX 10.1.5e* the condition is recorded in the Windows Application Event Log and in the trace log files when the runtime starts on Windows. In earlier versions it was discarded with no record at all. On a container the startup command line in the container log remains the direct check, as described below.

Confirm it from the startup command line in the container log, with `docker logs <container>` on each node.

Pair formed:

```
/solution:"/Documents/FrameworX/Solutions/Plant1.dbsln" /profile:0 /onlineConfig:true
/redundancy /ip1:172.17.0.2 /ip2:10.0.0.59 /port1:3101 /port2:3101
/connectiontimeout:3 /tstartup
```

Pair not formed, node running standalone:

```
/solution:"/Documents/FrameworX/Solutions/Plant1.dbsln" /profile:0 /onlineConfig:true
/port1:3101 /connectiontimeout:3 /tstartup
```

If `/redundancy`, `/ip1` and `/ip2` are absent, the configured addresses are not local to the container. Correct the configuration using one of the options below.

Option 1: Host Networking

The simplest configuration for two containers on two separate hosts. The container shares the host network stack, so the host addresses are local to the runtime and the ordinary redundancy configuration applies with no change.

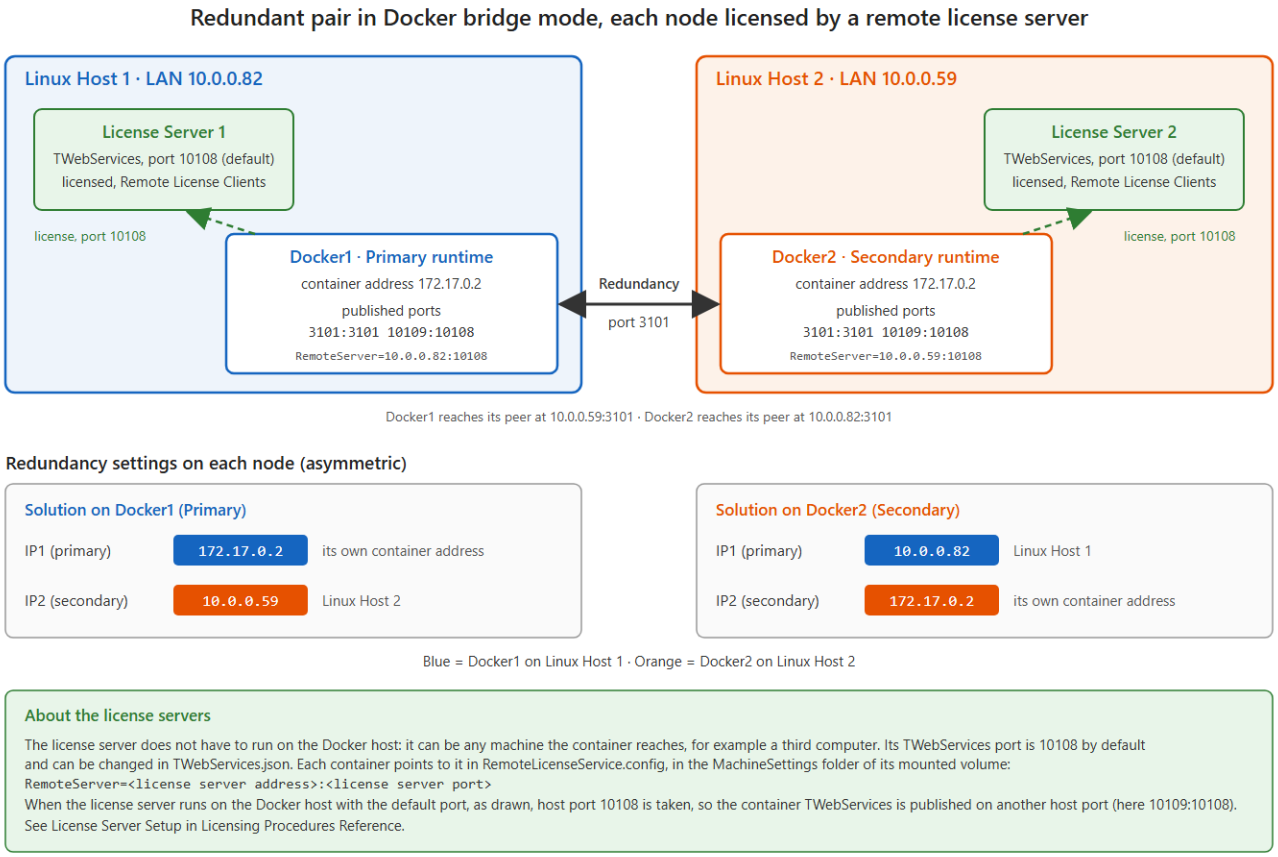
```
docker run -d --name fx-primary --network host \
  --mount type=bind,source=/opt/fx_data,target=/Documents \
  tatsoftdockerhub/frameworx:10.1.5
```

Configure IP1 and IP2 as the two host addresses, identically on both nodes. Host networking is Linux only.

Option 2: Bridge Networking Across Two Hosts

When host networking is not available, a pair can still be formed in bridge mode. Each node must reference **itself by an address that is local to its own container**, and the **peer by an address at which the peer is reachable**, which on a separate host is the peer host address with the runtime port published.

The configuration is therefore asymmetric. The two nodes do not carry the same pair of values.



The license server of each node can be its Docker host, as drawn, or any other computer the container reaches. When it runs on the Docker host with the default port 10108, that host port is taken, so publish the container TWebServices on another host port, for example `10109:10108`. See License Server Setup.

Node	IP1 (primary)	IP2 (secondary)
Primary, on host A (`10.0.0.82`), container address `172.17.0.2`	`172.17.0.2` its own container address	`10.0.0.59` host B
Secondary, on host B (`10.0.0.59`), container address `172.17.0.2`	`10.0.0.82` host A	`172.17.0.2` its own container address

Read the table by role. On the primary node, IP1 is the slot that identifies that node, so it holds an address local to that container. On the secondary node, IP2 is the slot that identifies that node, so it holds an address local to that container. The remaining slot always holds the peer host address.

Publish the runtime port on both containers so each host can reach the other:

```
services:
  frameworx:
    image: tatsoftdockerhub/frameworx:10.1.5
    ports:
      - "3101:3101"
      - "10108:10108"
    volumes:
      - ./fx_data:/Documents
```

Any firewall between the two hosts must allow the published runtime port in both directions.

Both Containers on One Docker Host

When the pair runs as two containers on a single host, create a user-defined bridge with fixed container addresses and configure those addresses as IP1 and IP2 on both nodes. The configuration is symmetric in this case, because both addresses live on the same Docker network and each is local to its own container.

```
docker network create --subnet 172.20.0.0/16 fxred

docker run -d --name fx-primary   --network fxred --ip 172.20.0.5 -p 3101:3101 ...
docker run -d --name fx-secondary --network fxred --ip 172.20.0.4 -p 3102:3101 ...
```

Set IP1 to `172.20.0.5` and IP2 to `172.20.0.4` on both nodes. A user-defined bridge does not span Docker hosts, so this pattern applies to a single-host pair only.

Verification Checklist

Run these on each node before concluding that a pair is configured correctly.

Step	Command	What to look for
1. Read the container network mode and address	`docker inspect -f '{{.HostConfig.NetworkMode}} {{range $k,$v := .NetworkSettings.Networks}}{{$k}}={{$v.IPAddress}} {{end}}' <container>`	The address reported here is the only address local to the runtime in bridge mode.
2. Compare with the solution configuration	Redundancy page of the solution on that node	The slot for this node's own role must hold the address from step 1.
3. Confirm the pair actually formed	`docker logs <container>`	`/redundancy`, `/ip1` and `/ip2` present on the startup command line.
4. Confirm reachability	From each host, open the peer runtime port	The published port answers from the other host.

Related Pages

- Redundancy Reference for the full redundancy model, the startup parameter list, and synchronization behavior.
- Docker Deployment Guide for image layout, persistent volumes, and general container deployment.

In this section...