

The manual

Version 1.0.0 · 2026-09-08 · docs.pathmri.com

CONTENTS

- 1. What PathMRI is [Getting started](#)
- 2. Install and first run [Getting started](#)

What PathMRI is

PathMRI answers one question well:

Can this reach that, on this port — and if not, exactly where does the packet die?

It answers with a real packet rather than a model, and it can do so without installing anything on the machines it tests. Around that answer it builds the things you need once the answer is *no*: which firewall's policy denies the flow, what NAT rewrote the addresses on the way, whether the return path comes back a different way, whether the name even resolves to the address you tested, and whether any of it changed since yesterday.

How it answers

Every check is three steps, and the third is the one that matters.

- 1. **Predict.** PathMRI reads the policy that governs the flow — the firewall's own configuration, a cloud security group, a Kubernetes network policy — and works out what it says should happen, naming the rule that decides.
- 2. **Probe.** It sends a real packet from inside the source host, and records what actually happened: reached, refused, or dropped, and where.
- 3. **Reconcile.** It compares the two. Agreement is a verified path. **Disagreement is a finding** — policy says allow and the wire says no, which is the case that costs teams days.

A port scanner only ever does step 2, from wherever the scanner happens to sit.

The pieces

Piece	What it is
Controller	The application you install. It serves the console in your browser, holds the database, and reaches out to hosts and devices. One process, no external services.
Agent	An optional small binary on a target host. It can run as a service, or be pushed for a single check and deleted afterwards.
Network devices	Read-only connections to your firewalls, routers and cloud control planes. PathMRI reads their policy; it does not need to change it.

The words the console uses

Host and network device. A **host** is something you verify *from* or *to* — a machine, a container, a pod. A **network device** is something *on the path*: a firewall, router or switch, whose policy PathMRI reads and which it can name as the place a packet

stopped. A device can be the far end of an ad-hoc check, traced *to* its management address, but it never originates a probe.

Management address and scan address. The **management address** is how the controller reaches a host to run a check. The **scan address** is the address other machines use to reach that host's services — so it is the address firewall policy is evaluated against, and the one probes are aimed at. They differ whenever a host sits behind a load balancer or a NAT. Leave the scan address blank and the management address is used for both.

Connection mode and connection type. The mode is *what runs on the host*; the type is *how the controller talks to it*. They are separate choices:

Mode	What runs on the host
Agent	A small service that starts with the machine. Fastest, and can measure everything. Removable at any time.
Agentless	Nothing. Checks run over SSH or WinRM using tools the host already has.
Ephemeral	A throwaway agent is pushed over SSH, runs in memory, and is deleted when the check ends. Agent-grade depth, nothing left on disk.

Type	How the controller connects
ssh	OpenSSH — the agentless path, and the tunnel a managed agent is reached through
agent	Straight to the agent's own port
winrm	WS-Management, for Windows fleets that have WinRM but not OpenSSH
docker	The host is a Docker engine
kube	The address is a kubeconfig context on the controller — no node SSH needed

Declared and observed. A **declared** answer comes from configuration: what a policy says should happen. An **observed** answer comes from a live state table on the device: what it is doing right now. PathMRI keeps the two apart everywhere, and never invents the observed half — where a platform does not expose one, it says so rather than implying a measurement.

Connectivity rule. A rule is an expectation you write down: *these sources should reach these destinations on these ports*. Rules are what a verification run executes, what a scheduled re-check re-runs, and what drift is measured against.

What a result means

"Blocked" is not one thing, and the difference decides who fixes it.

Result	What it means
open	A real service answered.
open verified	End to end confirmed — the destination's own listener received the probe. The path is clear even with no service running.
closed	The packet arrived and was actively refused. Nothing is blocking it, so this is deliberately not reported as a block.
filtered	Silence. Something is dropping packets, and the path view shows where.
app reset	The port is open and the application layer was reset — the signature of a UTM or application-layer block on an otherwise allowed flow.
app silent	The handshake completed and nothing came back at all. Typically a stateful firewall that never saw the reply.
error	The probe could not run. This is never reported as closed.

Every blocked flow is also attributed to a side: the source host's own firewall, the destination's, a firewall on the path, an intermediate device, or asymmetric routing — where nothing is denying it at all and the return leg simply goes another way.

What it will not do

- **It will not guess a vendor.** Tell PathMRI a device is something it does not recognise and it answers *unknown*, quoting the string it did not understand — rather than reading a device with the wrong parser and being confidently wrong.
- **It will not present an unmeasured path as closed.** A check that could not run is reported as unmeasured. This distinction is the difference between evidence and an assumption.
- **It will not claim more than the transport can see.** Each way of reaching a host measures a different amount, and a result says what was and was not established.

Where to go next

- **Install and first run** — from a package to a first verified path.
- **Reaching a host** — the three modes, and the credentials each one needs.
- **Firewalls and cloud policy** — what PathMRI reads on each platform.

Install and first run

From a downloaded package to a first verified path. Nothing is installed on the machines you test.

Pick a package

Platform	File	How to run it
Windows	PathMRI-Setup-win-x64.exe	Installs to Program Files, with a Start-Menu entry and an uninstaller. The usual choice.
Windows, portable	PathMRI-win-x64-portable.zip	Unzip, run <code>Start PathMRI.cmd</code> . Nothing is written outside the folder and your data directory, and no administrator rights are needed.
Linux	PathMRI-linux-x64.tar.gz	<code>./pathmri.sh</code> , or install the bundled <code>pathmri.service</code> systemd unit.
macOS	PathMRI-osx-x64.tar.gz, PathMRI-osx-arm64.tar.gz	<code>./pathmri.command</code> . It clears the Gatekeeper quarantine flag, because the binaries are unsigned.
Docker	the container image	<code>docker run -p 8080:8080 -e PATHMRI_MASTER_KEY=... -v pathmri-data:/data</code>

Every package is self-contained. There is no runtime to install first — the .NET runtime and the agent binaries for every platform are already inside.

Start it

A package you launch yourself opens your browser at the right address once the server is listening. That is suppressed automatically for a Windows service, a systemd unit, a container, or any process with redirected input. Turn it off explicitly with `--no-browser` or `PATHMRI_NO_BROWSER=1`.

Where it listens. `http://localhost:5180` natively, and `http://0.0.0.0:8080` in the container and under the shipped systemd unit. Change it with `PATHMRI_URL`; because the address is passed explicitly, it overrides `ASPNETCORE_URLS`.

Where its data lives.

Platform	Directory
Windows	<code>%APPDATA%\PathMRI</code>
Linux, systemd	<code>/var/lib/pathmri</code>
Container	<code>/data</code>
Anywhere	<code>\$PATHMRI_DATA_DIR</code> , if you set it

That directory holds the database, your settings, the data-protection keys, and — unless you supply `PATHMRI_MASTER_KEY` yourself — the key that protects stored credentials.

Back up the key with the database. Without it, the credentials in that database cannot be decrypted, and there is no recovery path. This is the one irreversible mistake available here.

First login

`admin / admin`, and the first thing it asks for is a new password.

The change is enforced by the server, not just by the screen: until it is done, the account can read who it is, set a password, and sign out. Everything else answers 403, including the in-browser shell. `admin/admin` stops working the moment a real password is set.

If you lock yourself out of the only administrator account, recover on the machine the controller runs on:

```
PathMRI.Api --reset-admin # back to admin/admin, with the forced change
PathMRI.Api --set-admin-password 'a new password'
```

Either one edits the database and exits. The server never starts, so neither can be used remotely.

Add the first host

Hosts & Servers → **+ Add hosts** takes five routes: type one in by hand, sweep a range, paste a list, import a CSV, or pull computer objects from Active Directory. For a first run, add one host by hand.

You need two things: an address the controller can reach it on, and a way in — SSH credentials, WinRM credentials, or nothing at all if you plan to deploy an agent later. See [Reaching a host](#) for which to choose and what each can measure.

Two columns on that screen answer different questions, and neither can be read off the other:

- **Agent** — whether a managed agent is installed and answering, with its version.
- **Health** — whether the machine itself is reachable, from a background check every 45 seconds.

A host with no agent can be perfectly healthy, and an installed agent can sit on a machine that is down.

The SSH key column shows the host key PathMRI has pinned. The first connection records whatever the host presents. If it ever presents a different one, the connection is refused **before your credential is sent**, and both fingerprints are shown so you can compare them against your own change record.

Run the first check

Two ways, and the second is faster the first time.

From the Unified Map. Click a source, click a destination, add the ports, and run. There is no form to fill in and no syntax to remember, and firewalls appear on the map as things you can select rather than gaps between two hosts.

From Run Verification. The ad-hoc panel checks any two endpoints with no rule at all: add a port, pick TCP, UDP or ICMP, add as many as you need, and run them in order. The activity feed stays on screen after the run, because the steps it lists are the evidence for the verdict.

Open a finished row and you get the verdict, the round-trip time, and the path: source, the hops between, destination, with firewalls marked along it.

Turn it into something that repeats

An ad-hoc answer is true for one minute. A **connectivity rule** is the same question written down — *these sources should reach these destinations on these ports* — and it is what scheduled runs re-execute and what drift is measured against. Any result worth keeping can become a rule from the result itself.

Where to go next

- **Reaching a host** — agent, agentless or ephemeral, and what each needs.
- **Firewalls and cloud policy** — connect your firewalls so a blocked port names the rule that stopped it.
- **Security and operations** — roles, backup, and where every secret is kept.