WhiteStar Dashboard Installation Guide

WhiteStar Dashboard is a self-hosted Node.js application. It's distributed as a single zip archive containing a prebuilt server and a start script that checks prerequisites, installs dependencies on first run, checks for updates, and launches the server. The same archive is used on Windows, macOS, and Linux — only the start script you run differs.

Download: https://whitestar.io/download/core/dashboard

Installing and starting the Dashboard

Windows

```
curl -L https://whitestar.io/downloads/dashboard/dashboard.zip -o dashboa
tar -xf dashboard.zip -C dashboard
cd dashboard
powershell -ExecutionPolicy Bypass -File start.ps1
```

`start.ps1` checks for Node.js (installing it with `winget` / `choco` if missing), verifies the version, runs `npm install` on first launch, checks for a newer `dashboard.zip` on the server, and then starts the server.

`-ExecutionPolicy Bypass` is scoped to this one process only — it does not change your system's PowerShell execution policy.

macOS / Linux

```
curl -L https://whitestar.io/downloads/dashboard/dashboard.zip -o dashboa
unzip dashboard.zip -d dashboard
cd dashboard && bash start.sh
```

On Linux and macOS, `start.sh` performs the same checks as `start.ps1` : Node.js version, first-run `npm install` , and an update check against the server before starting.

Accessing the Dashboard

Once you see 🚀 `Starting server...` in the terminal, the server is listening on port **3000** by default, bound to all network interfaces (`0.0.0.0`):

- From the same machine: http://localhost:3000
- From another device on the same network: `http://<machine-ip-address>:3000`

To use a different port, set the `PORT` environment variable before running the start script:

```
# macOS / Linux
PORT=8080 bash start.sh


# Windows
$env:PORT = "8080"
powershell -ExecutionPolicy Bypass -File start.ps1
```

Opening the port for LAN access

The Dashboard binds to all interfaces, but your OS firewall may still block inbound connections on port 3000 from other devices. You only need to do this if you want to reach the Dashboard from a device other than the one running it.

Windows

Run in an elevated (Administrator) PowerShell prompt:

```
New-NetFirewallRule -DisplayName "WhiteStar Dashboard" -Direction Inbounc
-Protocol TCP -LocalPort 3000 -Action Allow
```

macOS

macOS's Application Firewall is off by default, so no action is usually needed. If you've enabled it, add Node.js as an allowed app under **System Settings > Network > Firewall > Options...**, or allow the connection when prompted the first time the Dashboard is accessed remotely.

Linux

Fedora / RHEL-based (firewalld):

```
sudo firewall-cmd --permanent --add-port=3000/tcp
sudo firewall-cmd --reload
```

Debian / Ubuntu-based (ufw), if enabled:

```
sudo ufw allow 3000/tcp
```

Running the Dashboard as a background service (optional)

By default the Dashboard runs in the foreground of the terminal you started it in. If you want it to keep running after you close the terminal or reboot the machine, set it up as a service.

Linux (systemd)

Create `/etc/systemd/system/whitestar-dashboard.service`:

```
[Unit]
Description=WhiteStar Dashboard
After=network.target

[Service]
Type=simple
WorkingDirectory=/opt/whitestar-dashboard
ExecStart=/bin/bash /opt/whitestar-dashboard/start.sh
Restart=on-failure
RestartSec=5
Environment=HOSTNAME=0.0.0.0
Environment=PORT=3000

[Install]
WantedBy=multi-user.target
```

Adjust `WorkingDirectory` / `ExecStart` to match where you extracted the archive, then enable and start it:

```
sudo systemctl daemon-reload
sudo systemctl enable --now whitestar-dashboard.service
```

View logs with `journalctl -u whitestar-dashboard.service -f`. Because `start.sh`'s update prompt reads from stdin, and systemd services have no interactive terminal attached, the prompt is skipped automatically on each restart rather than hanging.

macOS (launchd)

Create `~/Library/LaunchAgents/io.whitestar.dashboard.plist`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.cc
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>io.whitestar.dashboard</string>
  <key>ProgramArguments</key>
  <array>
    <string>/bin/bash</string>
    <string>/Users/YOUR_USERNAME/whitestar-dashboard/start.sh</string>
  </array>
  <key>WorkingDirectory</key>
  <string>/Users/YOUR_USERNAME/whitestar-dashboard</string>
  <key>RunAtLoad</key>
  <true/>
  <key>KeepAlive</key>
  <true/>
  <key>StandardOutPath</key>
  <string>/tmp/whitestar-dashboard.log</string>
  <key>StandardErrorPath</key>
  <string>/tmp/whitestar-dashboard.err</string>
</dict>
</plist>
```

Replace `YOUR_USERNAME` and the install path, then load it:

```
launchctl load ~/Library/LaunchAgents/io.whitestar.dashboard.plist
```

Windows (Task Scheduler)

1. Open **Task Scheduler** and choose **Create Task...**.
2. General tab: name it "WhiteStar Dashboard", and select **Run whether user is logged on or not**.
3. Triggers tab: **New...** → **Begin the task: At startup**.

4. Actions tab: **New...** → Action **Start a program**:
    - Program/script: `powershell.exe`
    - Add arguments: `-ExecutionPolicy Bypass -File "C:\path\to\dashboard\start.ps1"`
    - Start in: `C:\path\to\dashboard`
5. Save the task, providing your account credentials when prompted.

Updating

Every time you run `start.sh` or `start.ps1`, it checks for a newer version and, if one is available, offers to download and install it in place before starting. To update, just re-run the start script.

Troubleshooting

- **"Node.js version X is not supported"** — install/upgrade to Node.js 22.12.0 or later, then re-run the start script.
- **Port 3000 already in use** — set the `PORT` environment variable to an unused port, as described above.
- **Can't reach the Dashboard from another device** — confirm the server is running with `HOSTNAME=0.0.0.0` (the default), and open port 3000 (or your custom port) in the firewall as described above.
- `start.ps1` **won't run / "running scripts is disabled on this system"** — use the `-ExecutionPolicy Bypass` flag shown above, which applies only to that single invocation.
