

pebble.

The User Guide

Turn a small FreeBSD box into a calm, well-behaved router. One YAML file. No drama.

Brain-friendly edition. Short sections. Big signposts. Every chapter opens with a **TL;DR**, says roughly how long it takes, and ends with a **Done when** check, so you always know when you can stop. Skip about freely: nothing here punishes you for reading out of order.

Start here: pick your path

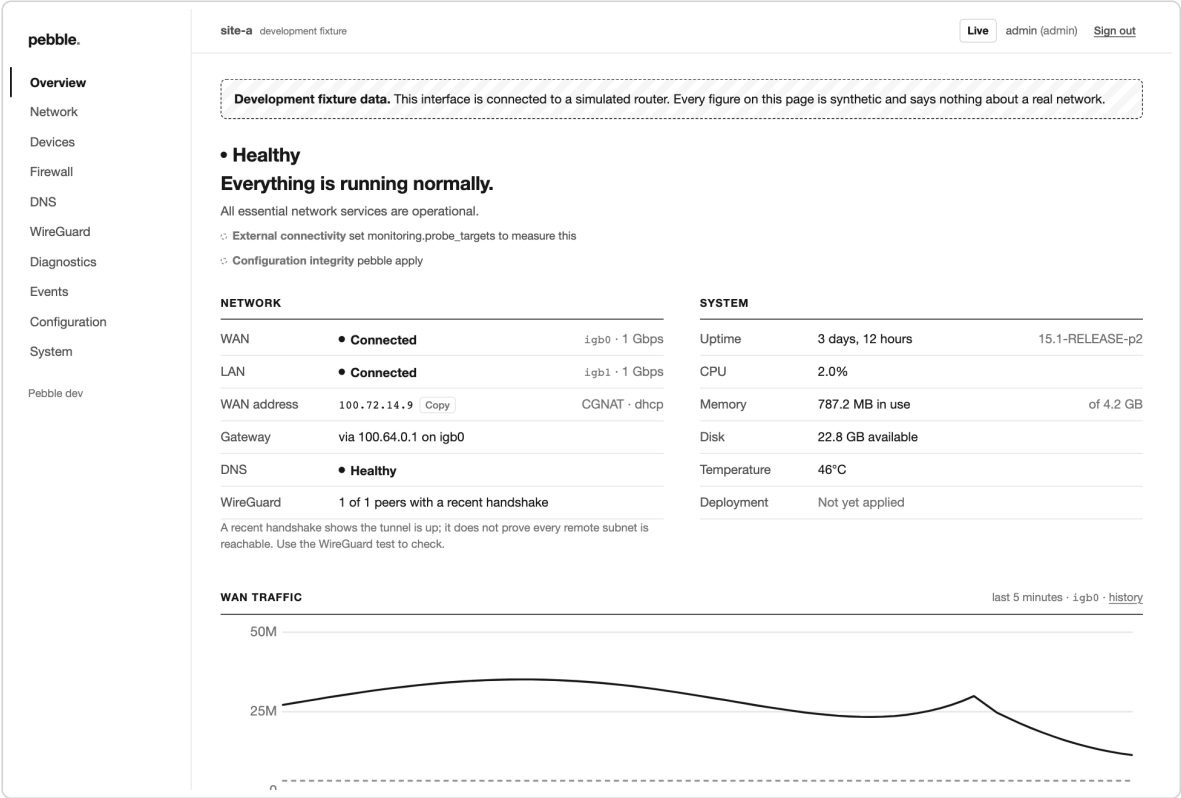
You do not need to read this whole guide. Find the row that matches you and jump.

I want to...	Go to	Time
Understand what Pebble actually is	Chapter 1	1 min
Build a router from a fresh FreeBSD install	Chapter 3	20 min
Change something without breaking the Internet	Chapter 5	5 min
Do a common job (reserve an IP, forward a port, block a device, use a VPN...)	Chapter 6	2 min each
Join two sites with WireGuard	Chapter 7	30 min
Work out why the Internet died at 3 a.m.	Chapter 8	5 min
Use the web interface	Chapter 9	10 min
Fix something that's wrong <i>right now</i>	Chapter 11	—
Just the commands, please	Chapter 12	—

The key, so you can scan

Marker	Means
TL;DR	The whole section in three lines. Read only this if you're in a hurry.
	Roughly how long it takes
Done when	How you know you've finished and can go and make tea
Tip	Nice to know. Safe to skip.
Careful	Read this one. It's here because someone got bitten.
Honesty box	What has and hasn't been tested. No spin.
Screenshots	Taken from Pebble running against a <i>simulated</i> router. The dashed "Development fixture data" banner is there on purpose: every number in them is synthetic.

Honesty box. Pebble is young. Its logic is heavily unit-tested and its generated configuration has been checked with the real Unbound and dnsmasq tools, but **it has not yet been run end-to-end on a real FreeBSD router**. Try it on spare hardware first, not on the only router between your family and Netflix. The full status table is in the project README.



📷 The web overview: four questions answered at a glance. (Simulated router.)

Contents

Start here: pick your path	3
The key, so you can scan	3
1 Pebble in 60 seconds	7
The mental model (this is the important bit)	7
What Pebble looks after	8
What Pebble is <i>not</i>	8
2 Before you begin	11
Checklist	11
3 Install: from nothing to router	13
Step 1: get the binary on to the box	13
Step 2: find out which port is which	13
Step 3: bootstrap (the big one)	13
Step 4: check, then confirm	14
4 A quick tour of pebble.yaml	15
5 The golden loop: changing things safely	17
The loop, step by step	17
Which changes need confirming?	17
The safety net, explained in one breath	17
Changed your mind?	18
6 Recipes: the everyday stuff	19
7 Site-to-site WireGuard	25
The idea	25
Step 1: keys	25
Step 2: the tunnel (on site A)	25
Step 3: say what may cross	26
Step 4: apply and test	26

8 Seeing what's going on	29
The quick look	29
The live dashboard	29
The flight recorder (optional, and brilliant)	30
9 The web interface	33
Switch it on	33
What's in there	34
Making changes in the browser	37
Two kinds of account	38
10 Backups, recovery and upgrades	41
Back up	41
Restore	41
Upgrade Pebble	41
11 Something's wrong	43
12 Cheat sheet	45
13 Glossary (jargon, decoded)	47

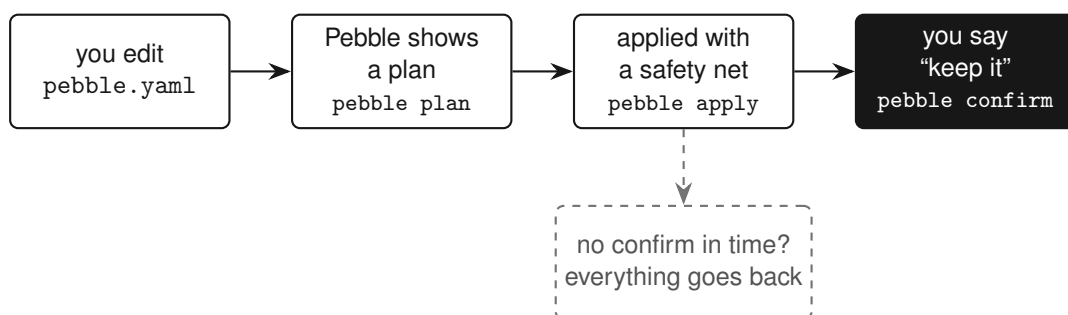
CHAPTER 1

Pebble in 60 seconds

⚡ TL;DR

- You describe your router in **one file**: `pebble.yaml`.
- Pebble turns that into **normal FreeBSD configuration** and applies it **safely**.
- Then it **gets out of the way**. FreeBSD does the routing, not Pebble.

The mental model (this is the important bit)



That's it. That's the whole product. Everything else is detail.

In the web interface, the *plan* step is a review page showing exactly what will change:

pebble. site-a development fixture admin (admin) [Sign out](#)

Overview
Network
Devices
Firewall
DNS
WireGuard
Diagnostics
Events
Configuration
System
Pebble dev

Development fixture data. This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.

Review changes

Disable firewall rule 2

- **Valid** The configuration compiles and passed the pre-flight checks.

live checks skipped: not running on FreeBSD

CONFIGURATION DIFF

+ enabled: false

DEPLOYMENT PLAN

ACTION	FILE	COMPONENT
Modify	/usr/local/etc/pebble/active/pf.conf	firewall

► /usr/local/etc/pebble/active/pf.conf

This change can interrupt connectivity (interfaces, routing or firewall). Pebble arms automatic rollback first: unless you confirm within 2m, the previous generation is restored by the router itself, even if this browser or the web service goes away.

► Proposed pebble.yaml

APPLY

Reason (recorded with the generation)
Disable firewall rule 2

Rollback protection
Configured default (2m) ▼

Apply **Cancel**

📷 Review before apply: the diff, the files that change, and a warning that this one can interrupt connectivity.

What Pebble looks after

- 🌐 **WAN** (DHCP or static) and **one LAN**, with NAT
- 🛡️ **Firewall** (PF): deny by default, your rules in order, port forwards
- 💡 **DHCP** (dnsmasq): addresses, reservations
- 🔍 **DNS** (Unbound): your own resolver, DNSSEC, local names, ad blocking
- 🗑️ **WireGuard**: routed site-to-site tunnels, and VPN provider exits (ProtonVPN, Mullvad. . .) for just the devices you choose
- 👤 **Per-device Internet**: each device goes out via the WAN, a VPN, or not at all
- ⚙️ **Safe changes**: every change can be undone, automatically if needed
- 🛠️ **Optional extras**: a flight recorder, a terminal dashboard and a web interface

What Pebble is *not*

- Not OPNsense or pfSense. Deliberately smaller.
- No cloud, no telemetry, no account with anyone.

- No IPv6 yet, no VLANs, no multi-WAN, no PPPoE.

💡 **Tip.** “Pebble gets out of the way” is literal. Delete the `pebble` binary and your router keeps routing, filtering and handing out addresses. Everything it set up is ordinary FreeBSD.

✅ **Done when** you could explain *edit* → *plan* → *apply* → *confirm* to a friend down the pub.

CHAPTER 2

Before you begin

TL;DR

- One amd64 box with **two network ports**, running **FreeBSD 15.1**.
- A **keyboard and screen** plugged in for the first setup.
- Ten minutes of nobody needing the Internet.

 5 minutes to check

Checklist

- ☐ An amd64 machine with **two Ethernet ports**. A Fujitsu FUTRO S930 thin client is the reference box; most mini PCs and VMs are fine.
- ☐ **FreeBSD 15.1-RELEASE** installed (a plain install is perfect).
- ☐ **Local console** access: a real keyboard and screen, or a VM console. Not SSH. You'll see why in a moment.
- ☐ You know which port will face the **Internet (WAN)** and which faces **your network (LAN)**.
- ☐ You've picked a LAN address range. 192.168.10.1/24 is the default and is fine for most homes.

 **Careful.** The very first deployment replaces the firewall and the network settings. Do it from the **local console**. If you do it over SSH and something's wrong, you're locked out until the automatic rollback kicks in (two minutes by default), which feels like a very long time.

 **Done when** every box above is ticked.

CHAPTER 3

Install: from nothing to router

⚡ TL;DR

1. Copy the pebble binary onto the box.
2. Run `pebble bootstrap` with your two ports.
3. Run `pebble confirm`. You have a router.

🕒 *About 20 minutes, most of it waiting for packages*

Step 1: get the binary on to the box

On any machine with Go installed, from the Pebble source:

```
make build-freebsd # makes bin/pebble-freebsd-amd64
```

Copy it across (USB stick, scp, carrier pigeon) and make it executable:

```
chmod +x pebble
```

Step 2: find out which port is which

```
./pebble interfaces
```

You'll see names like `igb0`, `igb1`, `em0` or `re0`.

💡 **Tip.** Not sure which is which? Unplug one cable, run the command again, and see which one changed to “no carrier”. Pebble will **never guess** for you, on purpose.

Step 3: bootstrap (the big one)

From the **local console**, swapping in your own ports, address, name and time zone:

```
./pebble bootstrap --wan igb0 --lan igb1 --lan-address 192.168.10.1/24 \
--hostname pebble --timezone Europe/London --apply
```

This one command:

- checks the machine is suitable
- installs **Unbound** and **dnsmasq**
- loads the kernel modules for the firewall and WireGuard
- installs Pebble itself, its service scripts and examples

- creates `pebble.yaml` from your answers
- applies it, with the safety net switched on

Want to see what it *would* do first? Add `--dry-run`. Already got the packages? Add `--skip-packages`. No Internet on the box? Use `--packages /path/to/pkg`.

Step 4: check, then confirm

Plug a laptop into the LAN port. Does it get an address? Can it open a website? Then:

```
| pebble confirm
```

 **Careful.** You've got **2 minutes** to confirm. Don't confirm, and Pebble quietly puts the machine back exactly as it was. That's a feature, not a bug.

✔ **Done when** a laptop on the LAN can browse the web and `pebble status` says everything's fine.

CHAPTER 4

A quick tour of pebble.yaml

⚡ TL;DR

- It lives at `/usr/local/etc/pebble/pebble.yaml`.
- One section per job: network, dhcp, dns, firewall, wireguard...
- Unknown or misspelt settings are **errors**, not silently ignored.

🕒 5 minutes

Here's a complete, working home router. Read the comments; that's the tour.

```
version: 1

system:
  hostname: pebble
  timezone: Europe/London

network:
  wan:
    device: igb0          # the Internet side
    ipv4:
      mode: dhcp          # or: static, with address and gateway
  lan:
    device: igb1          # your side
    ipv4:
      address: 192.168.10.1/24
    ipv6:
      mode: disabled      # IPv6 isn't supported yet

dhcp:
  enabled: true
  authoritative: true
  start: 192.168.10.100   # addresses handed out to devices
  end: 192.168.10.199
  lease_duration: 12h
  domain: home.arpa      # devices become name.home.arpa

dns:
  enabled: true
  recursive: true        # ask the root servers directly, no ISP in the middle
  dnssec: true           # check signatures
  cache_mb: 32
  blocking:
    enabled: true         # ad and tracker blocking (add lists later)
    response: nxdomain
    update_interval: 24h
    max_domains: 250000
    lists: []
    allowlist: []
    denylist: []
```

```
wireguard:
  interfaces: []

firewall:
  default_inbound: deny    # nothing from the Internet gets in...
  default_forward: deny    # ...or through, unless a rule says so
  default_outbound: allow
  nat:
    enabled: true
    outbound: wan

management:
  ssh:
    enabled: true
    allow_from:
      - 192.168.10.0/24    # SSH only from your LAN

updates:
  automatic_install: false

recovery:
  rollback_timeout: 120s   # how long you get to confirm
  keep_generations: 5      # how many old versions to keep for rollback
```

💡 **Tip.** Every single setting, with an explanation, is in `/usr/local/share/examples/pebble/annotated.yaml`. Keep it open in another window.

✅ **Done when** you can find your LAN address and DHCP range in the file without searching.

CHAPTER 5

The golden loop: changing things safely

⚡ TL;DR

```
pebble plan --diff    # look
pebble apply          # do
pebble confirm        # keep
```

🕒 1 minute per change, once it's a habit

The loop, step by step

1. **Edit** `/usr/local/etc/pebble/pebble.yaml`.
2. **Look before you leap** with `pebble plan --diff`. This changes *nothing*. It shows every file that would change, line by line, and warns you about anything risky.
3. **Apply** with `pebble apply`. Pebble validates everything with FreeBSD's own tools, takes a snapshot, applies it and runs health checks.
4. **Confirm** (if asked) with `pebble confirm`.

Which changes need confirming?

Change	Needs pebble confirm?
Firewall rules, port forwards	Yes
Interfaces, addresses, routes	Yes
WireGuard	Yes
DNS records, blocklists	No, if health checks pass
DHCP reservations	No, if health checks pass

Want a confirmation step for everything? `pebble apply --confirm-always`. Need longer than two minutes? `pebble apply --rollback-after 5m`.

The safety net, explained in one breath

If you don't confirm in time, because SSH dropped, the firewall locked you out, the power went or you simply wandered off, **the previous working configuration comes back on its own**. Three independent things make sure of it: a timer, a check every minute, and a check at boot before the network even starts.

In the web interface, a pending change looks like this on every page:

The screenshot shows the Pebble web interface for a development fixture. At the top, there's a header with 'pebble.' and 'site-a development fixture'. On the right, there are links for 'Live', 'admin (admin)', and 'Sign out'. A sidebar on the left lists navigation options: Overview, Network, Devices, Firewall, DNS, WireGuard, Diagnostics, Events, Configuration, System, and Pebble dev.

The main content area features a 'Development fixture data' banner stating: 'This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.' Below this is a yellow warning box: 'Generation 1 is pending. Unless it is confirmed, Pebble will restore the configuration from before Pebble's first deployment at 16:13:26 (in 1 min 43 s). The rollback runs on the router and does not depend on this browser.' It includes buttons for 'Confirm and keep', 'Roll back now', and 'Details'.

Below the warning, a 'Degraded' status is shown: 'The network is working, with problems.' It notes 'Attention needed: Configuration integrity.' and provides a link to 'External connectivity set monitoring.probe_targets to measure this'. A note indicates 'Configuration integrity' can be confirmed or rolled back.

Two tables are displayed:

NETWORK		
WAN	• Connected	igb0 · 1 Gbps
LAN	• Connected	igb1 · 1 Gbps
WAN address	100.72.14.9	CGNAT · dhcp
Gateway	via 100.64.0.1 on igb0	
DNS	• Healthy	
WireGuard	1 of 1 peers with a recent handshake	

A recent handshake shows the tunnel is up; it does not prove every remote subnet is reachable. Use the WireGuard test to check.

SYSTEM		
Uptime	3 days, 12 hours	15.1-RELEASE-p2
CPU	2.0%	
Memory	787.2 MB in use	of 4.2 GB
Disk	22.8 GB available	
Temperature	46°C	
Deployment	Generation 1	Pending

At the bottom, there's a 'WAN TRAFFIC' section with a graph showing 50M traffic over the last 5 minutes.

The countdown banner. The rollback runs on the router, not in your browser.

Changed your mind?

```
pebble rollback           # undo the pending change, or go back one version
pebble rollback --generation 3 # go back to a specific version
pebble status             # see which generation you're on
```

Tip. Every successful apply is a numbered **generation**. They're like save points in a game. Pebble keeps the last five by default.

Done when “plan, apply, confirm” happens without you thinking about it.

CHAPTER 6

Recipes: the everyday stuff

⚡ TL;DR

Each recipe is: **add this YAML** → **run the golden loop**. That's every recipe. Pick the one you need and ignore the rest.

🕒 2 minutes each

🔑 Give a device a fixed address

```
dhcp:
  reservations:
    - name: nas
      mac: "02:00:5e:00:53:10"
      address: 192.168.10.20
      hostname: nas
```

Bonus: `nas.home.arpa` now resolves on your network. To find a device's MAC address, look at `pebble dhcp leases`.

🔍 Add a local DNS name

```
dns:
  records:
    - name: printer.home.arpa
      type: A
      value: 192.168.10.30
```

Check it: `pebble dns test printer.home.arpa`

pebble.

Overview
Network
Devices
Firewall
DNS
WireGuard
Diagnostics
Events
Configuration
System
Pebble dev

site-a development fixture

admin (admin) [Sign out](#)

Development fixture data. This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.

DNS

Unbound resolver and DNS filtering.

RESOLVER

Unbound	• Running
Answering	• Yes
Mode	Full recursion
DNSSEC	• Validating

CACHE

Queries	184466	since Unbound started
Cache hits	153106	
Cache misses	31359	
Mean recursion	—	

Aggregate counters only, Pebble does not record which names are looked up.

TEST A NAME

Domain name

Type

nas.site-a.home.arpa

A

Test

Local domain inside site-a.home.arpa; names not listed locally return NXDOMAIN

Result **NOERROR, DNSSEC-validated.**

Answer nas.site-a.home.arpa. 300 A 93.184.215.14 NOERROR · 23ms · DNSSEC validated

FILTERING

DNS filtering is disabled (dns.blocking.enabled).

FEED	DOMAINS	LAST SUCCESS	STATE
No feeds configured.			

► Add a feed

 DNS, Test a name: every local policy that applies, then the real answer.

Forward a port to a device

```

firewall:
  port_forwards:
    - description: Web server on the NAS
      protocol: tcp
      external_port: "8443"
      target: 192.168.10.20
      target_port: "443"
      sources: [198.51.100.0/24]  # optional: who's allowed in

```

 **Careful.** A port forward opens a hole to the whole Internet unless you set sources. Only forward what you really mean to.

Stop a device reaching the Internet

Give the device a reservation (above) and add one line:

```

dhcp:
  reservations:
    - name: kids-tablet
      mac: "02:00:5e:00:53:60"
      address: 192.168.10.60
      egress: none  # no Internet; the LAN and router still work

```

Or, in the web interface: **Devices** → set *Internet via* to **Blocked** → **Change...** → review → apply. To undo, set it back to **WAN**.

💡 **Tip.** This rule is checked *before* your own firewall rules, so no pass rule can quietly let the device back out. The device keeps its old access until it renews its lease onto the reserved address (or you reconnect it).

🛑 Block an address on the Internet for everyone

To stop *every* device reaching one Internet address or network, use a firewall rule:

```
firewall:
  rules:
    - description: Nobody talks to this tracker
      action: block
      interface: lan
      protocol: any
      source: lan
      destination: 203.0.113.80      # or a prefix: 203.0.113.0/24
```

Swap source: `lan` for one address (192.168.10.30) to block just one device from just that destination. Rules are checked **top to bottom; the first match wins**. Want to switch a rule off without deleting it? Add `enabled: false`. To block by *name* rather than address, use the DNS denylist instead (see *Block ads and trackers* below).

⚠️ **Careful.** Devices sent through a VPN exit or blocked with egress skip these rules for their Internet traffic: the per-device setting always wins.

👤 Send one device through a VPN (ProtonVPN, Mullvad...)

Your VPN provider gives you a WireGuard configuration file. Copy its values into a WireGuard interface with role: `egress`, and save its PrivateKey value as `/usr/local/etc/pebble/keys/wg1.key` (mode 0600):

```
wireguard:
  interfaces:
    - name: wg1
      description: ProtonVPN UK
      role: egress          # only devices you send use it
      address: 10.2.0.2/32  # [Interface] Address
      gateway: 10.2.0.1    # [Interface] DNS
      dns: [10.2.0.1]      # [Interface] DNS
      private_key_file: /usr/local/etc/pebble/keys/wg1.key
      peers:
        - name: proton-uk
          public_key: <[Peer] PublicKey>
          endpoint: <[Peer] Endpoint>
          persistent_keepalive: 25
          allowed_ips: [0.0.0.0/0]

  dhcp:
    reservations:
      - name: laptop
        mac: "02:00:5e:00:53:21"
        address: 192.168.10.121
        egress: wg1          # Internet via the VPN
```

Or pick **VPN wg1** under *Internet via* on the **Devices** page. Everyone else keeps using your normal connection. What you get:

- **A kill switch.** If the tunnel drops, that device has *no* Internet, rather than quietly using your normal connection.
- **No DNS leaks.** The device is handed the VPN's DNS server when it next renews its lease.
- **Proof.** `pebble firewall test --in lan --src 192.168.10.121 --dst 1.1.1.1 --proto tcp --port 443` says the flow is passed by a `pebble-vpn-rule`.

 **Honesty box.** The firewall rules and their tests are done, but sending traffic through a WireGuard interface this way (`route-to`) has not been tried on a real FreeBSD router yet. Check that the device's public address is the VPN's before trusting it.

Block ads and trackers (DNS)

```
dns:
  blocking:
    enabled: true
    lists:
      - name: stevenblack
        url: https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts
        format: hosts
```

Apply, then fetch the list the first time:

```
pebble dns blocklist update
pebble dns blocklist test ads.example.com
```

After that it refreshes itself. A download that fails keeps the last good copy, so a broken list never breaks your DNS. Something you need got blocked? Add it to `allowlist`. Want to block a name yourself, with all its subdomains? Add it to `denylist` (for example `denylist: [tiktok.com]`).

Stop devices sneaking round your DNS (DNS-over-HTTPS)

Some apps ignore your DNS and talk to DoH servers directly. Block those servers by address:

```
firewall:
  ip_lists:
    update_interval: 24h
    lists:
      - name: doh
        url:
          https://raw.githubusercontent.com/dibdot/DoH-IP-blocklists/master/doh-ipv4.txt
    rules:
      - description: Block DNS-over-HTTPS bypass
        action: block
        interface: lan
        protocol: tcp/udp
        source: lan
        destination: list:doh
        ports: ["443"]
```

Then:

```
pebble firewall iplist update      # first download; it refreshes itself after
pebble firewall iplist test 1.1.1.1
```

🔗 “Would this be allowed?” (without trying it)

```
pebble firewall test --in lan --src 192.168.10.50 --dst 1.1.1.1 --proto tcp --
port 443
```

You get *pass* or *block* and the rule responsible. Nothing is sent over the network.

The screenshot shows the pebble. web interface. On the left is a sidebar with navigation links: Overview, Network, Devices, **Firewall**, DNS, WireGuard, Diagnostics, Events, Configuration, System, and Pebble dev. The main content area is titled 'site-a development fixture' and shows the Firewall configuration. A warning box states: 'Development fixture data. This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.'

Firewall
PF, managed entirely by Pebble.

PF	• Enabled	States	215 of 100,000	0.2%
Ruleset	• Pebble ruleset loaded	Rules loaded	17 filter · 1 translation	
Forwarding	• On	NAT	Outbound NAT on the WAN	

Pebble manages the whole PF ruleset (/usr/local/etc/pebble/active/pf.conf); administrator rules belong in /usr/local/etc/pebble/pf.local.conf (anchor pebble-local). IP forwarding is enabled only after PF has loaded Pebble's ruleset; there is deliberately no switch here that turns the firewall off while the router keeps forwarding.

Rules Policy summary **Test a flow** IP lists Generated pf.conf

TEST A FLOW
Evaluates a hypothetical packet against Pebble's model of the generated policy. Rules in the administrator anchor and existing connection states are not considered.

Arrives on	Source address	Destination	Protocol	Port
lan	192.168.10.50	1.1.1.1	tcp	443

Evaluate

Decision • **Pass** pass by pebble-lan-to-wan (LAN to the Internet (internal and site-to-site destinations need explicit rules))

Matching rule pebble-lan-to-wan LAN to the Internet (internal and site-to-site destinations need explicit rules)

Leaves via wan translated to the WAN address

Evaluated against Pebble's model of the generated policy; the administrator anchor and existing connection states are not considered.

📷 The same test in the web interface (Firewall, Test a flow).

✅ **Done when** your change is applied, confirmed (if asked) and works.

CHAPTER 7

Site-to-site WireGuard

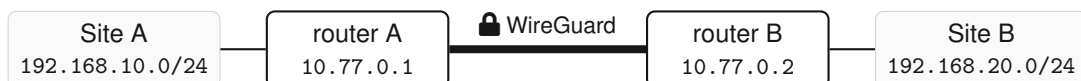
⚡ TL;DR

1. `pebble vpn genkey wg0` on each router.
2. Describe the tunnel and the other site in `pebble.yaml`.
3. Add firewall rules for what each side may reach. Apply. Test.

🕒 30 minutes, the first time

The idea

Two sites, each with its own LAN, joined by an encrypted tunnel. Traffic is **routed** (each side sees the other's real addresses) and **never NATed**. Nothing crosses the tunnel unless a firewall rule allows it.



Step 1: keys

On each router:

```
pebble vpn genkey wg0
```

This writes a **private key** to `/usr/local/etc/pebble/keys/wg0.key` (readable only by root) and prints the **public key**. Swap public keys between sites. Private keys never leave the box they were made on.

Step 2: the tunnel (on site A)

```
wireguard:
  interfaces:
    - name: wg0
      address: 10.77.0.1/24
      listen_port: 51820
      private_key_file: /usr/local/etc/pebble/keys/wg0.key
  peers:
    - name: site-b
      public_key: "PASTE-SITE-B-PUBLIC-KEY-HERE"
      endpoint: site-b.example.net:51820
      persistent_keepalive: 25
      allowed_ips: [10.77.0.2/32, 192.168.20.0/24]
      routes: [192.168.20.0/24]
```

Step 3: say what may cross

```
firewall:
  rules:
    - description: Site A LAN may reach all of site B
      action: pass
      interface: lan
      source: lan
      destination: peer:site-b
    - description: Site B may reach the file server over SMB only
      action: pass
      interface: wg0
      protocol: tcp
      source: peer:site-b
      destination: 192.168.10.5
      ports: ["445"]
```

Step 4: apply and test

```
pebble apply && pebble confirm
pebble vpn status
pebble vpn test site-b
```

pebble vpn test walks through the connection step by step: the peer's settings, the tunnel interface, the handshake, the route, the firewall, the traffic counters and a reply from the far side. It tells you **exactly which step failed**.

The screenshot shows the pebble. interface for site-a. The left sidebar contains navigation links: Overview, Network, Devices, Firewall, DNS, WireGuard (selected), Diagnostics, Events, Configuration, System, and Pebble dev. The main content area displays the WireGuard configuration for site-a, which is connected to a simulated router. The configuration includes:

- WireGuard**: Routed site-to-site tunnels. Inter-site traffic is never translated.
- wg0**: Interface up.
- Tunnel address**: 10.77.0.1/24
- Listening port**: UDP 51820
- Public key**: q0VcXw4dBqT3Hc8Jb6M0m0eVXaXk2jBg3dVbWmJ6PvW= (Copy)
- site-b**:
 - Handshake**: Recent (40 s ago)
 - Endpoint**: site-b.example.net:51820 (configured site-b.example.net:51820)
 - Transfer**: 22.4 MB received · 18.1 MB sent
 - Allowed IPs**: 10.77.0.2/32, 192.168.20.0/24
 - Remote networks**: 192.168.20.0/24 (route installed)
 - Local networks**: 192.168.10.0/24 (the remote side needs routes back to these)
 - Firewall**: 2 rules mention this peer
- Remote address to probe (optional)**: 192.168.20.0/24 (Run guided test)
- Remove peer...** button
- Add a peer to wg0** button
- RECENT TUNNEL EVENTS** section

📷 The WireGuard page: handshake, endpoint, transfer, routes and firewall coverage for each peer.

💡 **Tip.** Setting up the other end? `pebble vpn show site-b --endpoint your-router.example.net:51820` prints the settings site B needs.

⚠️ **Careful.** A fresh handshake means the tunnel is up. It does **not** prove that every remote network is reachable. That's what `pebble vpn test` is for.

✅ **Done when** `pebble vpn test site-b` passes every step.

CHAPTER 8

Seeing what's going on

⚡ TL;DR

- `pebble status`: a quick look.
- `pebble top`: a live dashboard in your terminal.
- `pebble explain outage`: what went wrong, and how sure Pebble is.

🕒 *Seconds*

The quick look

```
pebble status      # one-screen summary
pebble diagnose    # full health check, with evidence and what to do
```

Health is one of four honest words: **healthy**, **degraded**, **unavailable** or **unknown**. Pebble never shows “healthy” when something critical has failed, and never guesses when it doesn’t know.

The live dashboard

```
pebble top
```

Key	Does
1 2 3 4 or Tab	Overview, Interfaces, WireGuard, Events
p or space	Pause (handy for reading)
+ / -	Slower / faster refresh
↑↓ (or j k)	Scroll
q	Quit

Works happily over SSH, needs neither the web interface nor the agent, and has `--ascii` and `--plain` for awkward terminals.

```
ssh router — pebble top

pebble · FIXTURE DATA · site-a                                live · 2s · 16:05:27

● Healthy Everything is running normally.

NETWORK
WAN      ● up    igb0    1 Gbps  100.72.14.9/10    ↓ 27.0 Mb/s ↑ 3.3 Mb/s
LAN      ● up    igb1    1 Gbps  192.168.10.1/24   ↓ 3.3 Mb/s ↑ 27.0 Mb/s
DNS      ● answering
Firewall ● enabled 182 states of 100000 forwarding on
WireGuard 1 of 1 peers with a recent handshake

SYSTEM
Uptime    3 days, 12 hours  load 0.08 0.11 0.09
CPU        5.9%
Memory    787.2 MB of 4.2 GB
cpu0      46°C

RECENT EVENTS
No events recorded.
```

 pebble top over SSH. (Simulated router; note the FIXTURE DATA label in the header.)

The flight recorder (optional, and brilliant)

Switch it on once:

```
monitoring:
  enabled: true
```

From then on Pebble quietly records **events**: cables unplugged, driver hiccups, DHCP renewals, gateway changes, DNS failures, tunnel drops and configuration changes. It does **not** record your traffic or the websites you visit. Storage is capped (32 MB by default).

```
pebble events --last 24h           # the timeline
pebble events --severity warning --last 7d # just the worrying bits
pebble explain outage --last 1h     # the detective
```

`explain outage` sorts the evidence into four piles, so you always know how much to trust it:

Pile	Means
Observed	What actually happened, with times and sources
Likely explanation	What the sequence suggests
Not established	What the evidence <i>can't</i> tell you
Suggested next steps	What to try

pebble.

Overview
Network
Devices
Firewall
DNS
WireGuard
Diagnostics
Events
Configuration
System
Pebble dev

site-a development fixture

admin (admin) [Sign out](#)

Development fixture data. This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.

Events

The flight recorder: what this router observed, with its sources.

[Explain incidents](#)
[Export \(redacted\)](#)

INCIDENT ANALYSIS

17 Oct 2026 15:11 – 17 Oct 2026 16:11

Recorded events begin at 2026-10-17 16:08:47 BST; nothing before that can be examined.

No incidents found in this period.

Period
Type
Interface or subject
Search
Severity

1h
All
igb0
Any
Show

TIMELINE

4 events · last 1h · records since 17 Oct 2026 16:08

–1h

now

Saturday 17 October 2026

16:08:47

Recorder started

flight recorder started pebble

16:11:26

Configuration applied

config generation 1 applied (firewall); awaiting confirmation until 16:13:26 pebble

16:11:27

PF ruleset

pf ruleset 225bada78f42 loaded and verified monitor

16:11:43

Configuration confirmed

config generation 1 confirmed pebble

Journal: 1.3 kB of 33.6 MB in 1 segment(s). Pebble does not record DNS queries or traffic contents.

 The Events page: a timeline you can filter, plus *Explain incidents*.

Tip. A classic: “watchdog timeout, then link down, then link up”. Pebble will tell you the network card stopped sending, and will be honest that it can’t tell whether the driver, the card, the cable or the modem is to blame.

Done when you’ve run `pebble top` and pressed every number key at least once. (Go on. It’s fun.)

CHAPTER 9

The web interface

⚡ TL;DR

1. `pebble web user add yourname --role admin`
2. Add `management.web` to `pebble.yaml`, listening on your LAN address.
3. Apply, confirm, and browse to `https://192.168.10.1:8443/`.

🕒 10 minutes

It's **optional** and **off by default**. Routing never depends on it: stop it, and the router doesn't notice.

Switch it on

1. Make an account. There's **no default password**, ever.

```
| pebble web user add yourname --role admin
```

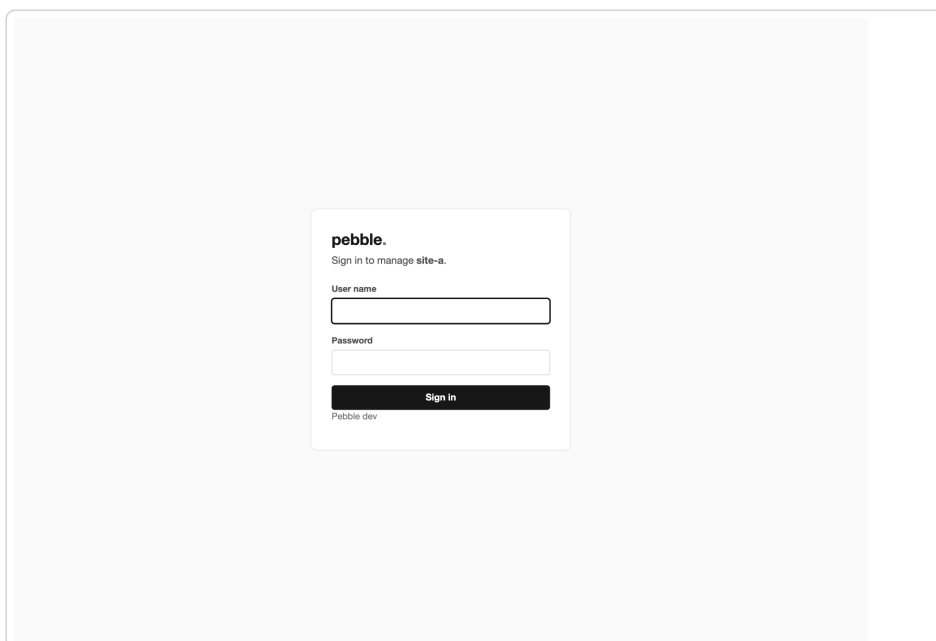
2. Add this under `management:` (next to `ssh:`), using your LAN address and range:

```
| management:
  web:
    enabled: true
    listen: 192.168.10.1:8443
    allow_from: [192.168.10.0/24]
```

3. The golden loop:

```
| pebble plan
| pebble apply
| pebble confirm
```

4. Open `https://192.168.10.1:8443/` in a browser.



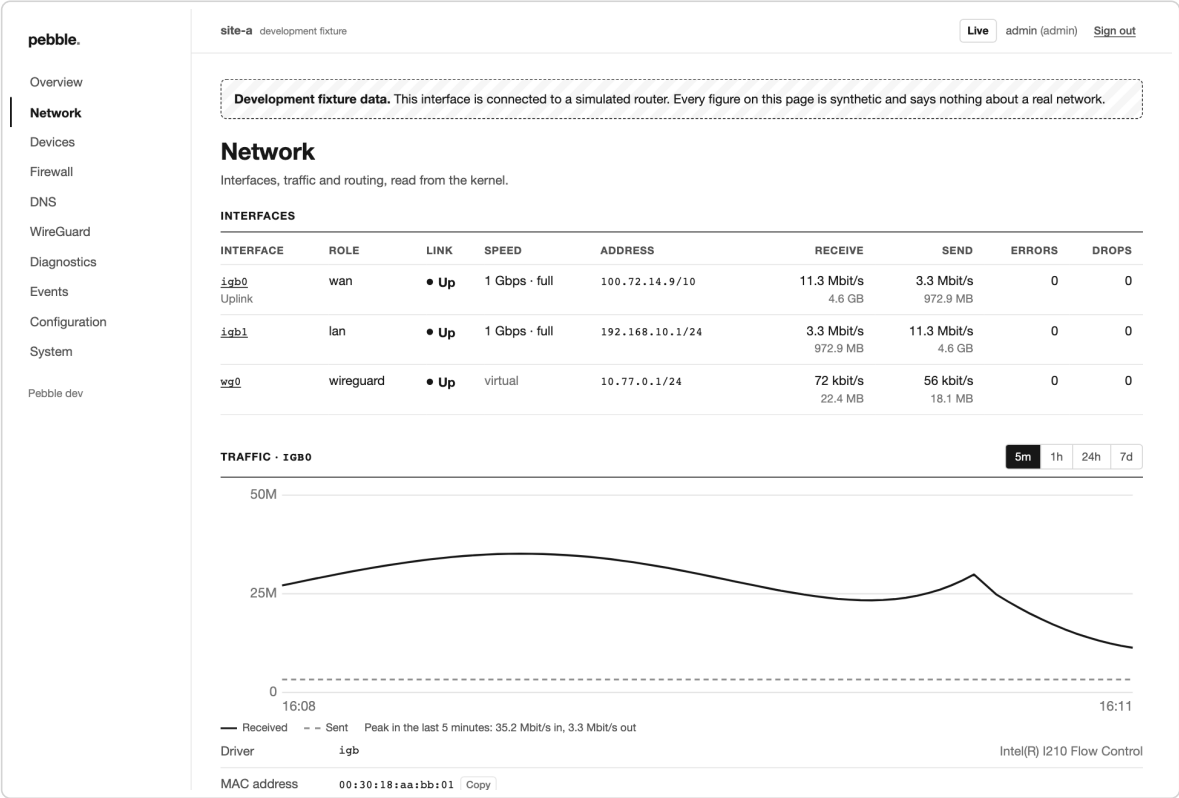
📷 No default password. Ever.

⚠ Careful. Your browser will warn about the certificate, because Pebble made its own. Run `pebble web cert` on the router and check that the **SHA-256 fingerprint** it prints matches the one your browser shows. Only then click through.

What's in there

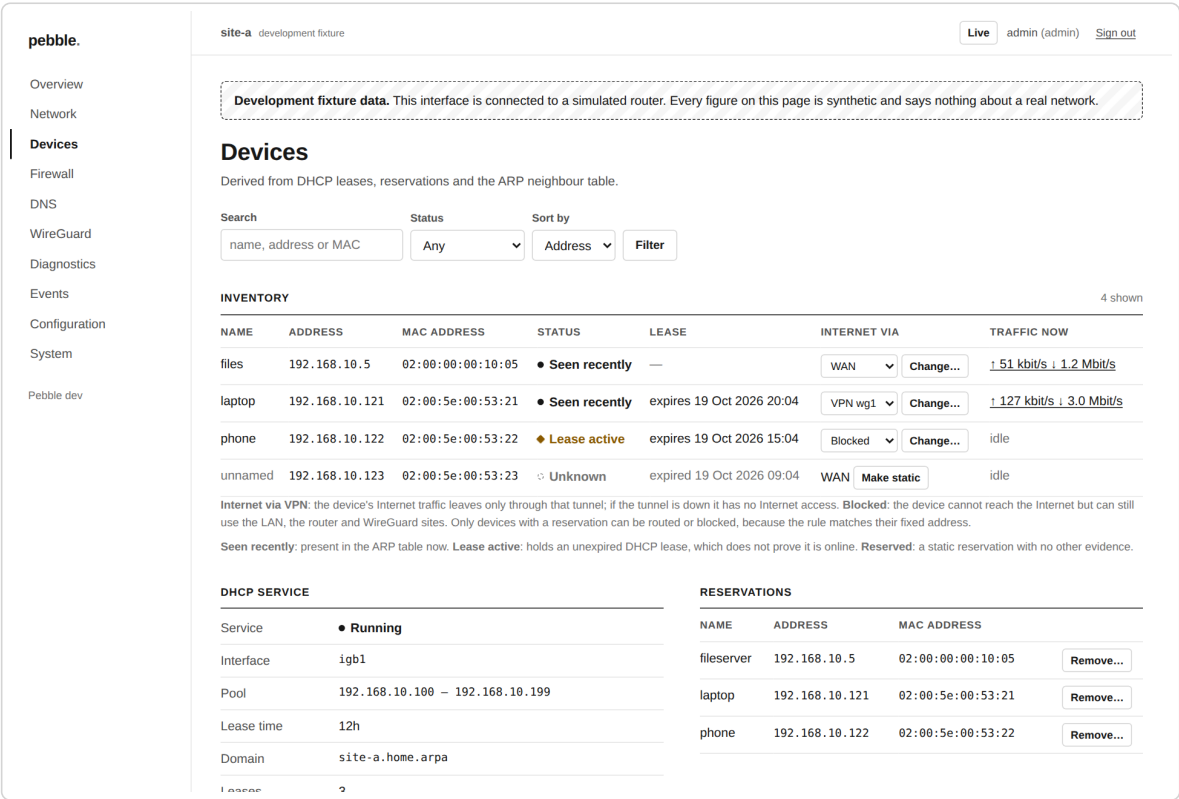
Page	What you get
Overview	Is the Internet OK? Are services OK? Anything odd? Anything to do?
Network	Interfaces, live traffic charts, routing table, “which route does this address use?”
Devices	Everything on your network, live traffic per device, and how each one reaches the Internet (WAN, VPN or blocked)
Firewall	Rules with match counts, policy summary, flow tester, generated <code>pf.conf</code>
DNS	Resolver health, test a name, blocklists, allow and deny lists, local names
WireGuard	Tunnels, peers, handshakes, guided connection test
Diagnostics	Health checks, ping, traceroute
Events	The flight recorder's timeline and incident explanations
Configuration	YAML editor, history, restore an older version
System	Hardware, services, storage, Pebble's own memory use

Network, with live traffic and the route lookup:



📷 Charts use black and grey with solid and dashed lines, so you never need colour to tell them apart.

Devices, with honest status words (“seen recently” is not the same as “lease active”):



Click a device’s traffic figures to watch its connections live:

pebble.

Overview

Network

Devices

Firewall

DNS

WireGuard

Diagnostics

Events

Configuration

System

Pebble dev

site-a development fixture

Live admin (admin) Sign out

Development fixture data. This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.

laptop

Live connections opened by this device, from the firewall's state table. [All devices](#)

DEVICE

Address192.168.10.121

MAC address02:00:5e:00:53:21

Status● Seen recently

Internet via● VPN wg1

NOW

Upload99 kbit/s

Download2.3 Mbit/s

Connections3

Rates over the last 2.2 s.

CONNECTIONS

3 shown, busiest first

PROTOCOL	REMOTE	LOCAL PORT	STATE	OPEN FOR	UP NOW	DOWN NOW	SENT	RECEIVED
UDP	192.168.10.1:53	50006	ESTABLISHED:ESTABLISHED	6 min	40 kbit/s	928 kbit/s	1.0 MB	23.6 MB
TCP	151.101.1.140:443	50005	ESTABLISHED:ESTABLISHED	5 min	33 kbit/s	773 kbit/s	841.5 kB	19.6 MB
TCP	1.1.1.1:443	50004	ESTABLISHED:ESTABLISHED	4 min	27 kbit/s	619 kbit/s	673.2 kB	15.7 MB

Each connection is counted where it enters the router, so traffic through a VPN exit is included. Traffic between devices on the LAN does not pass through the router and is not shown.

📷 Every connection the device has open, busiest first, refreshed every two seconds. Pebble keeps none of it: it is read from the firewall when you look.

Firewall, with match counters and lint warnings:

pebble.

Overview

Network

Devices

Firewall

DNS

WireGuard

Diagnostics

Events

Configuration

System

Pebble dev

site-a development fixture

admin (admin) Sign out

Development fixture data. This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.

Firewall

PF, managed entirely by Pebble.

PF● Enabled

Ruleset● Pebble ruleset loaded

Forwarding● On

States214 of 100,0000.2%

Rules loaded17 filter · 1 translation

NATOutbound NAT on the WAN

Pebble manages the whole PF ruleset (/usr/local/etc/pebble/active/pf.conf); administrator rules belong in /usr/local/etc/pebble/pf.local.conf (anchor pebble-local). IP forwarding is enabled only after PF has loaded Pebble's ruleset; there is deliberately no switch here that turns the firewall off while the router keeps forwarding.

Rules

Policy summary

Test a flow

IP lists

Generated pf.conf

RULES

evaluated in order; the first match wins

#	ACTION	INTERFACE	SOURCE	DESTINATION	PROTOCOL	PORTS	DESCRIPTION	MATCHES
1	● Pass	lan	lan	peer:site-b	any	any	Site A LAN may reach all of site B	343 51.7 kB
2	● Pass	wg0	peer:site-b	192.168.10.5	tcp	445	Site B may reach the file server over SMB only	343 51.7 kB

Match counts are PF label counters since the ruleset was last loaded. Every change opens a review showing the configuration diff and deployment plan.

► Add a rule

RECENT FIREWALL EVENTS

No events recorded.

📷 Every rule, its counters, and buttons to disable, move or remove it (each one goes through review).

Configuration, with the YAML editor and history:

36

The screenshot shows the pebble web interface. On the left is a sidebar with navigation links: Overview, Network, Devices, Firewall, DNS, WireGuard, Diagnostics, Events, Configuration (selected), System, and Pebble dev. The top header shows 'site-a development fixture' and 'admin (admin) Sign out'. The main content area has a dashed box with the text: 'Development fixture data. This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.' Below this is a message: 'The deployment was confirmed and is now permanent.' The 'Configuration' section shows 'pebble.yaml, its deployment history and the protected apply workflow.' The 'EDITOR' section contains a code editor with the following content:

```
# Site A of a routed WireGuard site-to-site VPN.
#
# Site A LAN 192.168.10.0/24, tunnel address 10.77.0.1
# Site B LAN 192.168.20.0/24, tunnel address 10.77.0.2
#
# Site B uses the mirror image of this file (see docs/wireguard.md).
# Create the key first: pebble vpn genkey wg0
version: 1

system:
  hostname: site-a
  timezone: Europe/London

network:
  wan:
    device: igb0
    ipv4:
      mode: dhcp
  lan:
    device: igb1
    ipv4:
      address: 192.168.10.1/24
    ipv6:
      mode: disabled

dhcp:
```

Below the editor is a 'Summary of the change' section.

 Edit the YAML directly; it still goes through review, plan and protected apply.

Making changes in the browser

Every change, whether from a button or the YAML editor, shows you a **review page** first: the exact lines changing in `pebble.yaml`, every generated file affected, and any warnings. Only then can you press **Apply**. Risky changes show a banner with a countdown and two buttons: **Confirm and keep** or **Roll back now**.

After you press **Apply**, you watch it happen:

pebble.

Overview

Network

Devices

Firewall

DNS

WireGuard

Diagnostics

Events

Configuration

System

Pebble dev

site-a development fixture

Updated 4:11:27 PM admin (admin) [Sign out](#)

Development fixture data. This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.

Generation 1 is pending. Unless it is confirmed, Pebble will restore the configuration from before Pebble's first deployment at 16:13:26 (in 1 min 58 s). The rollback runs on the router and does not depend on this browser.

Confirm and keep

Roll back now

Details

Apply

Started by admin at 16:11:25, finished at 16:11:26.

• Finished

If the deployment is pending, confirm it from the banner at the top of the page once you have checked that the network still works.

Applying generation 1

- set host name

\$ /bin/hostname site-a

- set time zone

\$ /usr/sbin/tzsetup -s Europe/London

- disable IPv6 forwarding

\$ /sbin/sysctl net.ipv6.forwarding=0

- load pf kernel module

\$ /sbin/kldload -n pf

- load PF ruleset

\$ /sbin/pfctl -f /usr/local/etc/pebble/active/pf.conf

- enable PF

\$ /sbin/pfctl -e

- enable IPv4 forwarding

\$ /sbin/sysctl net.inet.ip.forwarding=1

- add route 192.168.20.0/24

\$ /sbin/route -q add -net 192.168.20.0/24 -interface wg0

Running health checks

Configuration

Overview

 Every step, as it runs.

 **Tip.** The rollback timer runs on the **router**, not in your browser. Close the tab, lose Wi-Fi, drop your laptop in the bath: the router still puts things back if you don't confirm.

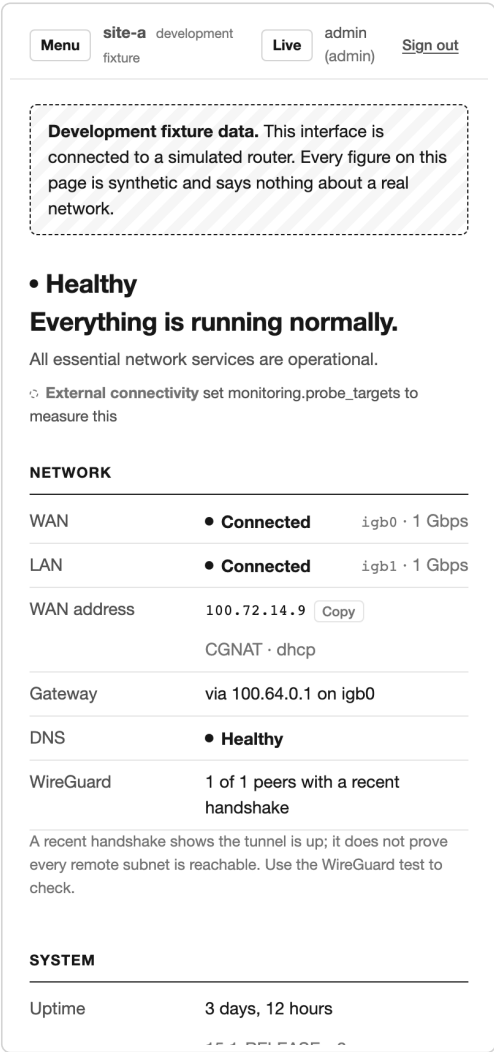
Two kinds of account

Role	Can
viewer	Look at everything, test things, preview changes
admin	All that, plus apply, confirm and roll back

```
pebble web user add guest --role viewer
pebble web user list
pebble web user passwd yourname
pebble web user remove guest          # their sessions end at once
```

 **Tip.** It works on your phone too.

38



 The overview on a phone-sized screen.

 **Done when** you can sign in and the Overview says your network is healthy.

CHAPTER 10

Backups, recovery and upgrades

⚡ TL;DR

- `pebble backup` now and then.
- If a change goes wrong, Pebble usually fixes it before you notice.
- Upgrade: new binary → `pebble bootstrap --skip-packages` → `pebble plan`.

🕒 2 minutes

Back up

```
pebble backup                # saved to /var/db/pebble/backups/
pebble backup --include-keys  # also WireGuard keys (keep this file safe!)
```

Copy the file somewhere that isn't the router.

Restore

```
pebble restore pebble-backup.tar.gz --dry-run  # see what it would do
pebble restore pebble-backup.tar.gz           # puts pebble.yaml back
pebble apply && pebble confirm                 # and deploy it
```

Upgrade Pebble

```
pebble update check          # tells you about updates; installs
    nothing
./pebble-new bootstrap --skip-packages  # installs the new binary and service
    scripts
pebble plan                  # see what the new version would change
```

💡 **Tip.** `pebble bootstrap` is safe to run as often as you like. It only does what's missing.

✅ **Done when** a backup file lives somewhere other than the router.

CHAPTER 11

Something's wrong

⚡ TL;DR

Breathe. Run `pebble diagnose`. It usually tells you what's wrong *and* what to do. Then find your symptom below.

🔥 The Internet is down

- 1. `pebble diagnose`: read the first thing marked **unavailable**.
- 2. `pebble interfaces`: is the WAN port **up**, with a link and an address?
- 3. `pebble explain outage --last 1h`: what happened, in order.
- 4. Just made a change? `pebble rollback`.

pebble.

Overview

Network

Devices

Firewall

DNS

WireGuard

Diagnostics

Events

Configuration

System

Pebble dev

site-a development fixture

admin (admin) [Sign out](#)

Development fixture data. This interface is connected to a simulated router. Every figure on this page is synthetic and says nothing about a real network.

Diagnostics

Health checks and network tools. Every result comes from this router.

Run full check

HEALTH CHECKS

local checks - the full check also queries upstream DNS

• Healthy Everything is running normally.

CHECK	STATUS	EVIDENCE	WHAT TO DO
WAN link critical Internet	• Healthy	igb0 carrier, 1 Gbps full	
WAN address critical Internet	• Healthy	100.72.14.9/10 (carrier-grade NAT range)	
Default route critical Internet	• Healthy	via 100.64.0.1 on igb0	
Upstream gateway Internet	• Healthy	100.64.0.1: 2/2 replies, average 8.430 ms	
External connectivity Internet	⚠ Unknown	no probe targets configured	set monitoring.probe_targets to measure this
LAN critical Network	• Healthy	igb1 192.168.10.1/24	
Firewall critical Security	• Healthy	enabled, ruleset e1e392fb80d1 verified, 211 states of 100000	
IPv4 forwarding critical Security	• Healthy	on, behind the verified ruleset	
DHCP Services	• Healthy	dnsmasq running on igb1	
DNS resolver critical	• Healthy	Unbound answering (0s)	

📷 Diagnostics in the web interface: every check, its evidence, and what to do about it.

🔒 I've locked myself out over SSH

Wait. If the change is unconfirmed, it rolls back by itself within the rollback window (2 minutes by default). Then reconnect and look at what you changed. If you'd already confirmed it, use the local console and run `pebble rollback`.

Websites don't load, but ping 1.1.1.1 works

That's DNS.

```
pebble dns test example.com      # what happened to this name, and why
pebble dns status
```

If a site you need is being blocked, add it to `dns.blocking.allowlist`.

A device didn't get an address

```
pebble dhcp status
pebble dhcp leases
```

Check the DHCP range isn't full, and that a reservation isn't pinning the device to a different address.

The tunnel's up but I can't reach the other site

Run `pebble vpn test site-b`. The usual culprits, in order:

1. A missing firewall rule on **one** side.
2. A missing **route back** on the other router.
3. `allowed_ips` that doesn't include the remote LAN.

A device has no Internet through its VPN

1. `pebble vpn status`: the VPN exit needs a recent handshake. None? Check the endpoint, the provider's public key and the private key file, all from the **same** provider file.
2. `pebble firewall test --in lan --src <device> --dst 1.1.1.1 --proto tcp --port 443` should name a `pebble-vpn-` rule. If not, the device isn't on its reserved address yet: reconnect it.

No Internet while the tunnel is down is the kill switch doing its job.

"Is my firewall even loaded?"

Run `pebble firewall status`. Pebble only enables forwarding once *its own* ruleset is loaded in PF. If it isn't, the router stops forwarding rather than running without a firewall.

 **Done when** the thing works again. (Then take a backup.)

CHAPTER 12

Cheat sheet

Job	Command
Look before changing	<code>pebble plan --diff</code>
Apply	<code>pebble apply</code>
Keep it	<code>pebble confirm</code>
Undo	<code>pebble rollback</code>
Go back to version N	<code>pebble rollback --generation N</code>
Check the file only	<code>pebble validate</code>
Quick status	<code>pebble status</code>
Full health check	<code>pebble diagnose</code>
Live dashboard	<code>pebble top</code>
What happened?	<code>pebble events --last 24h</code>
Why did it break?	<code>pebble explain outage --last 1h</code>
Network ports	<code>pebble interfaces</code>
Routing table	<code>pebble routes</code>
Which route?	<code>pebble route explain 192.168.20.10</code>
Firewall counters	<code>pebble firewall status</code>
Firewall in plain English	<code>pebble firewall explain</code>
Would it pass?	<code>pebble firewall test --in lan --src ... --dst ... --proto tcp --port 443</code>
Block a device's Internet	<code>reservation egress: none</code> (or <code>Devices</code> → <code>Blocked</code>)
Send a device via a VPN	<code>reservation egress: wg1</code> (or <code>Devices</code> → <code>VPN wg1</code>)
Who's using the bandwidth?	<code>web interface</code> → Devices → traffic figures
DHCP leases	<code>pebble dhcp leases</code>
Test a DNS name	<code>pebble dns test example.com</code>
Refresh blocklists	<code>pebble dns blocklist update</code>
Refresh IP lists	<code>pebble firewall iplist update</code>
VPN peers	<code>pebble vpn peers</code>
Test a tunnel	<code>pebble vpn test site-b</code>
New WireGuard key	<code>pebble vpn genkey wg0</code>
Web account	<code>pebble web user add name --role admin</code>
Web certificate fingerprint	<code>pebble web cert</code>
Back up	<code>pebble backup</code>
Import from OPNsense	<code>pebble import opnsense config.xml --map wan=igb0,lan=igb1</code>

Most commands take `--json` for scripts, and `-h` for help.

CHAPTER 13

Glossary (jargon, decoded)

Word	Plain English
WAN	The Internet-facing port
LAN	Your network's port
NAT	Lets all your devices share one Internet address
PF	FreeBSD's firewall
DHCP	Hands out addresses to devices automatically
Reservation	"This device always gets this address"
DNS	Turns names like <code>example.com</code> into addresses
DNSSEC	Checks DNS answers haven't been tampered with
Generation	A numbered, saved version of your router's configuration
Rollback	Going back to an earlier generation
Peer	The router at the other end of a WireGuard tunnel
Handshake	WireGuard's "hello, it's me" between peers
VPN exit	A WireGuard tunnel to a VPN provider that only chosen devices use
Kill switch	If the VPN drops, its devices get no Internet instead of leaking out the normal way
Egress	"Way out": how a device reaches the Internet
Agent	The optional background helper behind the web interface and flight recorder
Flight recorder	Pebble's log of important network events

**Open Pebble. Understand the network.
Make a change confidently. Close Pebble.
The router carries on quietly doing its job.**