

cpedgeOS Quick Start

Supported Boards

Board	Config	Description
rock-5b	config/boards/rock-5b.conf	Padxa Rock 5B (bootloader written to image)
symbiote	config/boards/symbiote.conf	Symbiote (vendor U-Boot in SPI NOR, ESP partition)

Build Commands

```
# Rock 5B - vendor kernel 6.1, Ubuntu 24.04
sudo ./build.sh rock-5b

# Rock 5B - mainline kernel 6.18 (Panther GPU + Rocket NPU)
sudo KERNEL_PROFILE=6.18 ./build.sh rock-5b

# Symbiote - vendor kernel 6.1
sudo ./build.sh symbiote

# Symbiote - mainline kernel 6.18
sudo KERNEL_PROFILE=6.18 ./build.sh symbiote

# Ubuntu 25.04 (Mesa Teflon from repos, no source build needed)
sudo UBUNTU_VERSION=25.04 KERNEL_PROFILE=6.18 ./build.sh rock-5b

# Use prebuilt U-Boot (skip U-Boot source build)
sudo ./build.sh --prebuilt-uboot rock-5b

# Run individual stages
sudo ./build.sh rock-5b kernel image
```

Host Build Dependencies

The mainline 6.18 kernel profile with KERNEL_PROFILE=6.18 cross-compiles Mesa Teflon on the host (not under QEMU). Ensure these are installed:

```
sudo apt install ninja-build python3-mako pkg-config \
    gcc-12-aarch64-linux-gnu g++-12-aarch64-linux-gnu
```

Meson >= 1.4.0 is auto-installed via pip3 if needed.

Output

Image: `../os_images/<kernel-ver>/<soc>/<board>-cpedgeos-<ubuntu-ver>-v<timestamp>.img`
Example: `symbiote-cpedgeos-24.04-v1771035186.img`

The v<timestamp> build ID is a Unix timestamp embedded in the filename, `/etc/os-release`, and shown in the login MOTD for version tracking.

Flash to SD Card

```
sudo dd if=rock-5b-cpedgeos-24.04-v1771035186.img of=/dev/sdX bs=4M status=progress sync
```

```
# Verify  
sha256sum -c rock-5b-cpedgeos-24.04-v1771035186.img.sha256
```

Replace `/dev/sdX` with your SD card device.

Flash to NVMe (via SPI)

Boot chain: Boot ROM -> SPL (SPI) -> U-Boot (SPI) -> kernel (NVMe)

1. Flash Radxa stock SPI image

The inindev prebuilt U-Boot does NOT support SPI NOR boot. Use the Radxa stock SPI image:

Download: <https://dl.radxa.com/rock5/sw/images/others/rock-5b-spi-image-gd1cf491-20240523.img>

From a running system on the Rock 5B:

```
# Erase SPI NOR flash  
sudo flash_erase /dev/mtd0 0 0  
  
# Write SPI image (do NOT use flashcp - hangs at 96% on 16MB images)  
sudo dd if=rock-5b-spi-image-gd1cf491-20240523.img of=/dev/mtdblock0 bs=4096 conv=fsync
```

2. Write OS image to NVMe

```
sudo dd if=rock-5b-cpedgeos-24.04-v1771035186.img of=/dev/nvme0n1 bs=4M status=progress sync
```

3. Kernel requirement

The NVMe driver must be built-in (`CONFIG_BLK_DEV_NVME=y`, `CONFIG_NVME_CORE=y`), not a module. The 6.18 kernel profile has this set. The default arm64 defconfig ships =m which won't work without an initramfs.

Default Credentials

Field	Value
User	cpedge
Password	cpedge
Sudo	Passwordless (NOPASSWD)
SSH keys	overlay/<board>/home/cpedge/.ssh/authorized_keys

Serial Console

Kernel	Device	Baud
6.1 (vendor)	ttyFIQ0	1500000
6.18 (mainline)	ttyS2	1500000

```
screen /dev/ttyUSBO 1500000
```

Login MOTD

On SSH login, the system displays a branded banner with live system stats:

- CPU model, core count (read from device-tree on mainline)
- Memory total / available
- Storage total / used / free
- CPU temperature
- Uptime, load average
- IP addresses
- Kernel version
- OS version and build ID (v<timestamp>)

What's on the Image

Kernel / Driver Stack

	6.1 (vendor)	6.18 (mainline)
GPU driver	Mali blob (libmali-valhall-g610)	Panthor (Mesa, open-source)
NPU driver	RKNPU2 (librknnrt.so)	Rocket (DRM accel, open-source)
NPU inference	rknn-toolkit-lite2	TFLite + libteflon.so (Mesa)
NPU device	/dev/rknpu	/dev/accel/accel0

Note: On 6.18 with Ubuntu 24.04, Mesa Teflon (`libteflon.so`) is cross-compiled on the host at native x86 speed using `aarch64-linux-gnu-gcc-12`. This avoids the slow QEMU-emulated build.

Virtualization

KVM, QEMU (`qemu-system-arm`), libvirt, LXC, nftables, bridge-utils

Network

- DHCP on all Ethernet interfaces matching `en*` (catches Realtek 2.5GbE as `enP4p65s0`)
- Renderer: `systemd-networkd`
- DNS: `systemd-resolved` with fallback to `8.8.8.8 / 1.1.1.1`
- Hostname: `cpedge-<mac>` (auto-generated from MAC address on first boot)

Node Registration Agent

Installed when NODE_AGENT_SRC is set in the board config (enabled for both rock-5b and symbiote).

- **Runtime:** Node.js 22.x LTS with npm (via NodeSource)
- **Location:** /opt/node-registration-agent/
- **Service:** node-registration-agent.service (enabled, auto-starts on boot)
- **Config:** /opt/node-registration-agent/.env (API URL, API key, polling interval)

```
# Manage the agent
sudo systemctl status node-registration-agent
sudo systemctl stop node-registration-agent
sudo systemctl start node-registration-agent
sudo journalctl -u node-registration-agent -f
```

Override the .env at build time with NODE_AGENT_ENV=/path/to/.env or edit on the device.

Boot Sequence

Image Disk Layout — Rock 5B

```
0      32 MiB          end of image
|--bootloader--|----- rootfs (ext4, GPT) -----
^           ^
|   +-- u-boot.itb @ sector 16384 (8 MiB)
+----- idbloader.img @ sector 64 (32 KiB)
```

- Single GPT partition (LABEL=rootfs), starts at 32 MiB
- U-Boot bootloader written into the gap before the partition
- fstab: LABEL=rootfs / ext4 defaults,noatime 0 1

Image Disk Layout — Symbiote

```
0    24 MiB  64 MiB          end of image
|--gap--|--ESP (40M)--|----- rootfs (ext4, GPT) -----
• ESP (EFI System Partition): sectors 49152–131071 (24 MiB – 64 MiB), FAT16, empty
• rootfs: starts at sector 131072 (64 MiB)
• No bootloader written to image — vendor U-Boot lives in SPI NOR
• The empty ESP triggers vendor U-Boot's distro boot detection
```

Boot Chain (SD / eMMC) — Rock 5B

RK3588 Boot ROM

```
→ idbloader.img (TPL + SPL, sector 64)
  → u-boot.itb (ATF + U-Boot proper, sector 16384)
    → extlinux.conf (/boot/extlinux/extlinux.conf)
      → kernel Image + DTB + cmdline
        → systemd (PID 1)
```

Boot Chain — Symbiote (SPI NOR)

RK3588 Boot ROM

- Vendor U-Boot in SPI NOR flash
- Finds ESP in GPT → triggers distro boot
 - extlinux.conf (/boot/extlinux/extlinux.conf)
 - kernel Image + DTB + cmdline
 - systemd (PID 1)

Boot Chain (NVMe via SPI)

RK3588 Boot ROM

- SPI NOR flash (Radxa stock image, /dev/mtdblock0)
- U-Boot (reads NVMe)
 - extlinux.conf on NVMe partition
 - kernel Image + DTB + cmdline
 - systemd (PID 1)

extlinux.conf (generated at image build time)

```
default linux-<version>
label linux-<version>
    kernel /boot/Image
    fdt /boot/rk3588-rock-5b.dtb
    append root=PARTUUID=<uuid> rootfstype=ext4 rootwait rw console=<tty>,1500000n8 console=tty1
    • root=PARTUUID=... — the GPT partition UUID, set during image assembly (no initramfs needed)
    • console=<tty> — ttyFIQ0 on 6.1, ttyS2 on 6.18
    • console=tty1 — HDMI output (both kernels)
```

U-Boot Source

Component	Repo	Branch
U-Boot	https://github.com/radxa/stable-5.10-rock5	boot.git
rkbin (BL31 + DDR)	https://github.com/rockchipster/linux/rkbin.git	
Prebuilt (inindev)	https://github.com/inindev/rk7-rc7	5b/releases

First-Time Startup

On the very first power-on after flashing, the system goes through these stages in order:

1. Rootfs Resize (early boot, ~2 min for 512 GB drive)

Service: `resize-rootfs-firstboot.service` (runs before `sysinit.target`)

The OS image is smaller than most SD cards or NVMe drives. On first boot, the system automatically:

1. Detects the root block device (`/dev/mmcblk0p1`, `/dev/nvme0n1p1`, etc.)

2. Runs `growpart` to expand the GPT partition to fill the disk
3. Runs `resize2fs` online to grow the ext4 filesystem
4. Writes marker `/var/lib/resize-rootfs/done` — won't run again

If it fails: Boot continues normally but disk space is limited to the image size. Fix manually:

```
sudo growpart /dev/mmcblk0 1      # or /dev/nvmeOn1 1
sudo resize2fs /dev/mmcblk0p1    # or /dev/nvmeOn1p1
```

2. Hostname Generation (first boot)

Service: `set-hostname-firstboot.service`

Sets a unique hostname from the primary Ethernet MAC address:

1. Waits up to 15 seconds for an `en*` network interface to appear
2. Reads the MAC address and generates `cpedge-<last-3-octets>` (e.g., `cpedge-032ae2`)
3. Writes to `/etc/hostname` and updates `/etc/hosts`
4. Marks done — won't run again

3. systemd Brings Up Services

Key services start in this order (approximate):

Target / Service	What it does
<code>sysinit.target</code>	Basic system init (after resize completes)
<code>systemd-networkd</code>	Configures Ethernet via netplan (<code>en*</code> → DHCP)
<code>systemd-resolved</code>	DNS resolver (stub at 127.0.0.53 + fallback 8.8.8.8)
<code>systemd-timesyncd</code>	NTP time sync
<code>serial-getty@<tty></code>	Login prompt on serial console
<code>getty@tty1</code>	Login prompt on HDMI
<code>ssh</code>	OpenSSH server (accepts key + password auth)
<code>node-registration-agent</code>	Node registration agent (reports to API)
<code>lxc-net</code>	LXC bridge network (<code>lxcbr0</code>)
<code>multi-user.target</code>	System fully up — triggers hw-test

Masked: `systemd-networkd-wait-online` — prevents boot from hanging if no DHCP server is reachable.

4. Hardware Test (after `multi-user.target`, ~10-30 min)

Service: `hw-test-firstboot.service`

Once the system is fully up, hw-test runs automatically in stress mode:

1. Detects all hardware (CPU, memory, GPU, NPU, storage, USB, network, thermal)
2. Runs functional and stress tests on each component

3. Generates HTML report at `/var/log/hw-test/hw-test-report.html`
4. Writes marker `/var/lib/hw-test/first-boot-complete` — won't run again

Timeout: 30 minutes. If tests hang, the service is killed by systemd.

Monitor progress on a running system:

```
# Follow live output
sudo journalctl -u hw-test-firstboot.service -f

# Check if still running
systemctl status hw-test-firstboot.service
```

5. System Ready

After hw-test completes, the system is idle and ready for use. Login via:

```
# Serial console
screen /dev/ttyUSB0 1500000

# SSH (once you know the IP)
ssh cpedge@<ip-address>

# Find IP from serial console
ip addr show
```

Subsequent Boots

On all boots after the first:

- Resize, hostname, and hw-test services skip (marker files exist)
- Boot to login prompt takes ~15-25 seconds (SD card) or ~10-15 seconds (NVMe)
- All services from the table above start normally

Running hw-test Manually

```
sudo hw-test --quick      # Detection only (seconds)
sudo hw-test --functional # Detection + functional tests (minutes)
sudo hw-test --stress     # Full stress tests (10-30 min, default)
sudo hw-test --report     # Also generate HTML report
```

What each mode tests:

- **CPU** — Core count, big.LITTLE topology, frequency scaling, stress-ng
- **Memory** — Size, stress-ng (60s)
- **GPU** — Render node detection, stress test
- **Storage** — FIO read/write (30s)
- **Network** — Ethernet detection, iperf3 throughput
- **USB** — Device enumeration
- **Thermal** — Temperature monitoring (120s, threshold 85C)
- **NPU** — MobileNet V1 inference (auto-detects Rocket or RKNPU driver)

Logs: `/var/log/hw-test/`

Key File Locations on the Image

Path	Purpose
/boot/Image	Kernel binary
/boot/rk3588-rock-5b.dtb	Device tree blob
/boot/extlinux/extlinux.conf	U-Boot boot config
/etc/netplan/01-netcfg.yaml	Network config (DHCP on en*)
/etc/fstab	LABEL=rootfs / ext4 defaults,noatime 0 1
/etc/os-release	cpedgeOS <version> branding + BUILD_ID
/etc/hostname	Hostname (set to cpedge-<mac> on first boot)
/etc/resolv.conf	Static DNS (127.0.0.53 + 8.8.8.8 + 1.1.1.1)
/etc/sudoers.d/cpedge	Passwordless sudo for cpedge user
/home/cpedge/.ssh/authorized_keys	Pre-installed SSH public keys
/usr/local/bin/hw-test	Hardware test suite
/usr/local/bin/resize-rootfs	First-boot partition resize
/usr/local/bin/set-hostname	First-boot hostname generation
/opt/node-registration-agent/	Node registration agent (Node.js)
/opt/node-registration-agent/.env	Agent config (API URL, key, interval)
/var/log/hw-test/	Test logs and HTML reports
/var/lib/resize-rootfs/done	Resize completion marker
/var/lib/set-hostname.done	Hostname set marker
/var/lib/hw-test/first-boot-complehehw-test completion marker	hw-test completion marker

NVMe Boot Troubleshooting

Problem	Fix
Board doesn't boot from NVMe	Flash Radxa stock SPI image (inindev U-Boot doesn't support SPI NOR)
flashcp hangs at 96%	Use dd if=... of=/dev/mtdblock0 bs=4096 conv=fsync instead
Kernel panic — can't find rootfs on NVMe	NVMe driver must be built-in (=y not =m); check CONFIG_BLK_DEV_NVME=y and CONFIG_NVME_CORE=y
MTD layout looks different	Mainline = single mtd0 (16MB); vendor/Mender = multiple partitions
Boot hangs with no network	systemd-networkd-wait-online is masked by default; if unmasked, it blocks boot without DHCP
No serial output on mainline kernel	Check console is ttys2 not ttymcu0 — mainline uses standard UART2

Problem	Fix
GPU probe fails (error -2) on mainline	Panthor must be =m (module), not =y (built-in) — firmware loads after rootfs mount
