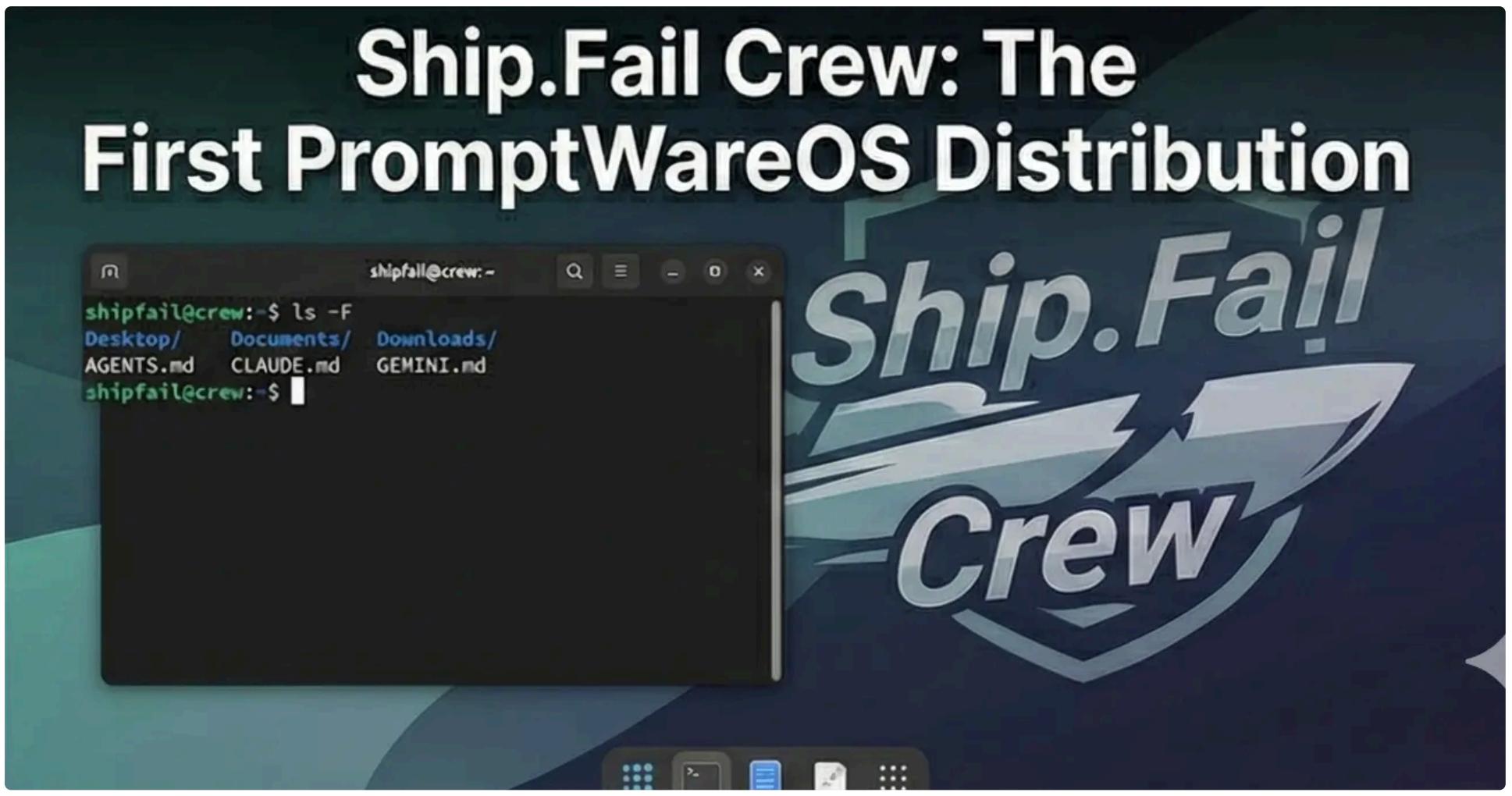
ENGINEERING

Ship.Fail Crew: The First PromptWareOS Distribution

Huan Li Follow

Dec 19, 2025 · 9 mins read



If you've ever installed Linux, you already know the punchline.

The **kernel** isn't the whole system. A kernel is the core: it enforces rules, exposes system calls, and keeps the machine honest. What makes Linux *usable* is everything wrapped around it: the shell, utilities, conventions, and packages—what we casually call a **distribution**.

PromptWareOS is built on the same separation.

- **PromptWareOS** provides the **boot protocol and kernel primitives**.
- Your repo provides the **worktree** (the code you're changing).
- And you need a third piece: a **userland** that defines your AI co-founders—agents, skills, and tool contracts.

That third piece is what we're shipping today: **Ship.Fail Crew**, the first PromptWareOS distribution.

Crew is designed to be mounted from the cloud as the AI's "home directory," so any project repo can boot into a consistent co-founder environment with **conventions alone**.

0) The pivot: prompts must ship with an operating environment

We're living through the "**English hits ring 0**" era.

Once words can drive tools, mount resources, route execution, and enforce policy, prompts stop being "text." They become **control planes**. In other words: **boot code**.

And boot code has a rule:

> *You don't ship boot code without an operating environment.*

So the real question isn't "what's the perfect prompt?"

It's: **how do we package the prompt *with* the environment it assumes?**

That's why PromptWareOS exists—and why it now needs distributions.

---

1) The problem: promptware becomes folklore when it lives everywhere

If you've built agent workflows for more than a week, you've probably felt this:

- You copy/paste the "best prompt so far" into a second repo.
- You tweak it for that repo.
- A month later, you have **five slightly different versions**.
- Nobody knows which one is "the real one."

Meanwhile the prompt keeps growing:

- tool usage rules
- conventions
- safety constraints
- project vocabulary
- "don't forget to do X"

The prompt becomes a suitcase you drag from repo to repo. Heavy. Fragile. Easy to lose.

So we accept a bad trade:

- either keep the prompt small and **lose guarantees**,
- or keep it huge and **waste context**.

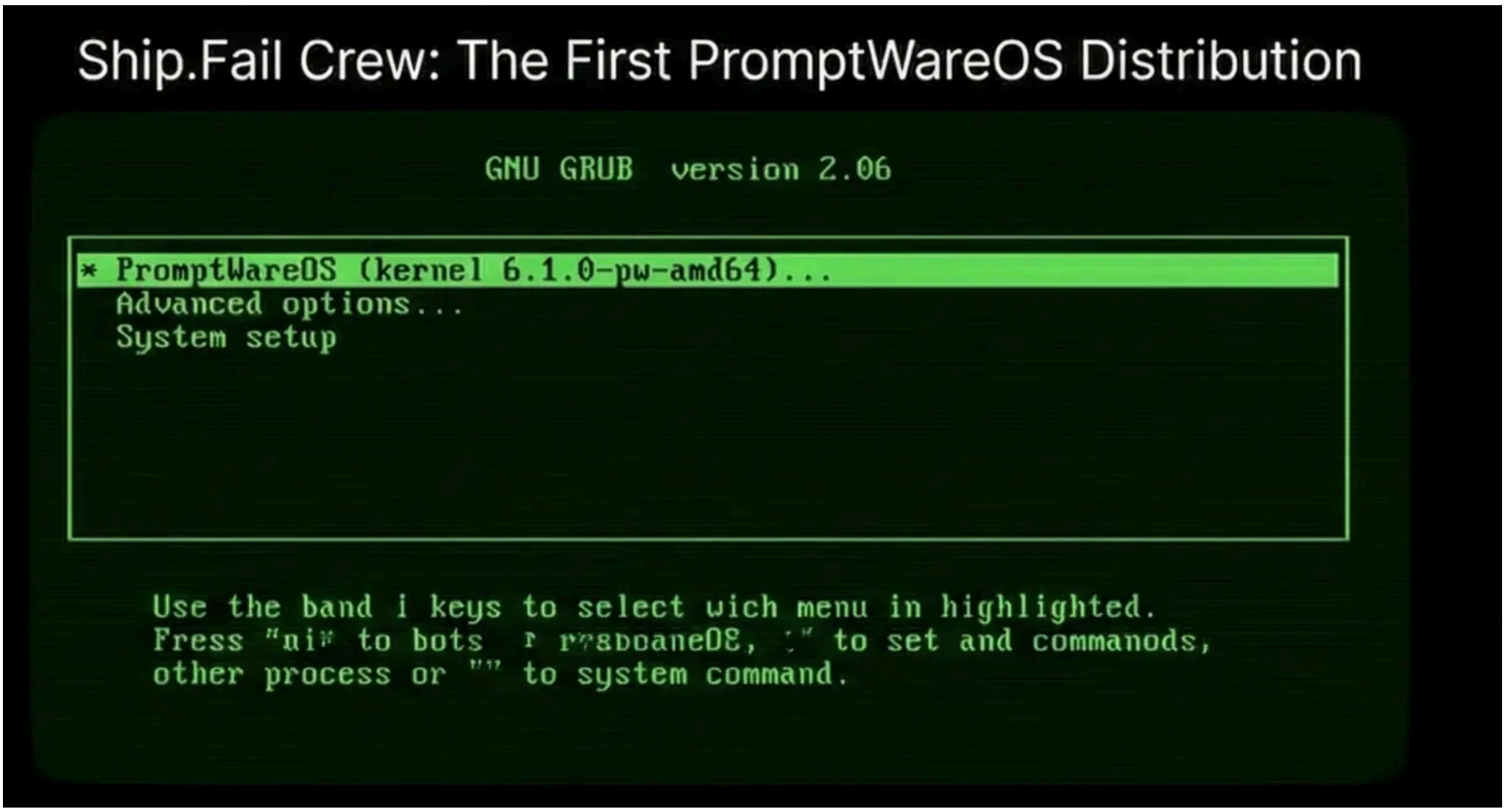
The fix is not “write better prompts.”

The fix is to treat promptware like software: **version it, mount it, and boot it**.

---

2) The OS move: make the bootloader immutable, mount everything else

PromptWareOS makes a hard separation that Linux people will recognize:



- **Bootloader** = read-only, authoritative: identity + topology + init
- **Runtime state** = mutable: memory, caches, working data

In PromptWareOS terms, the bootloader lives in your `AGENTS.md` as a small boot block.

It’s the anchor that says:

- “this is the root”
- “these are the mounts”
- “this is the init agent”

Everything else—skills, tools, and userland—is loaded on demand.

That gives you the most important property of an OS:

> *A reboot restores a clean world.*

If the agent gets confused mid-session, you don’t pray it “remembers.” You **reboot** into deterministic topology.

“

---

3) Mini-primer: Linux boot + userland in 3 minutes

If you’re new to Linux internals, here’s the mental model we’ll reuse.

Bootloader

The bootloader is the first serious decision-maker.

It decides **what system** you’re booting and **where to find it**.

- In many Linux setups, GRUB loads the kernel and passes boot parameters.
- The key idea: **it’s early and authoritative**.

Kernel

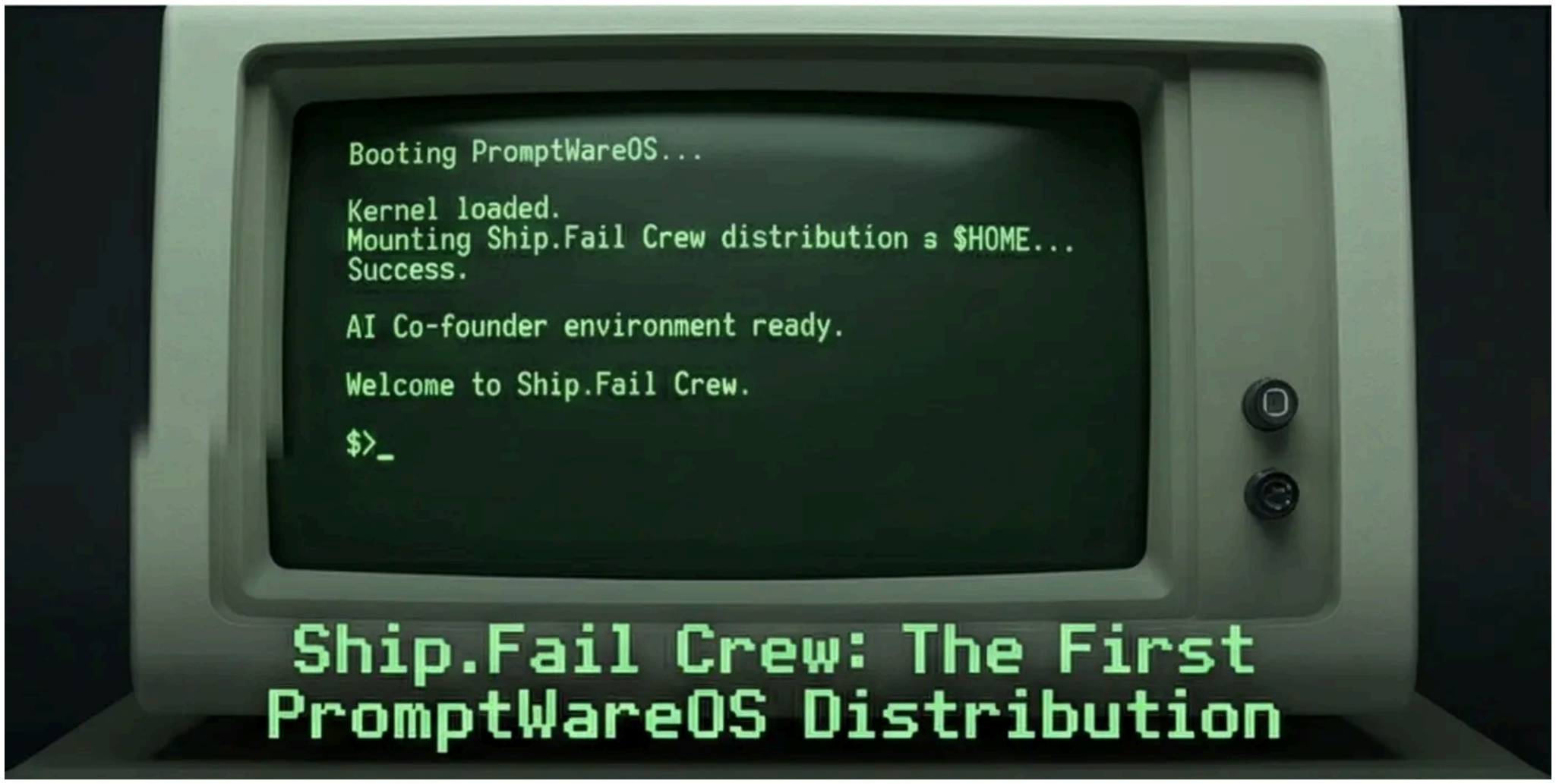
The kernel is the rulebook.

It provides:

- process scheduling
- memory management
- device abstraction
- system calls (the official “API” programs use)

Userland

Userland is everything you actually touch:



- shell (`bash`, `zsh`)
- core utilities (`ls`, `grep`, `cat`)
- package manager
- libraries
- programs

A Linux *distribution* is mostly userland plus policies about how it’s packaged.

`/etc/fstab`

`/etc/fstab` is a simple mount table.

It maps:

- **logical mount points** (like `/home`)
- to **physical devices/locations** (like a disk UUID)

Programs shouldn’t care which disk your `/home` lives on.

They just use `/home`.

Bridge: PromptWareOS swaps disks for URLs

PromptWareOS borrows the **shape** of Linux—but swaps:

- disks → URLs
- binaries → Markdown

- syscalls → tools/contracts

And the key trick is the same one: **mount logical paths to physical locations**.

---

4) The “aha”: PromptWareOS has the kernel. Now it needs a distro.

PromptWareOS decomposes promptware like an OS:

> ***bootloader → microkernel → init agent → skills → tools***

That gives us the stable base.

But a kernel alone is not what people run.

A kernel needs a **distribution**—a userland that ships:

- agents
- skills
- tool contracts
- conventions
- project overlays

So we’re introducing the first distribution for PromptWareOS:

Ship.Fail Crew

Crew is a mountable `$HOME` for AI co-founders.

If PromptWareOS is the kernel, then Crew is your **Debian**—your packaged userland.

---

5) The three-repo model (burn this into your brain)

Here’s the whole design in one diagram:

```
PromptWareOS kernel          Ship.Fail distro              Your project
(shipfail/promptware)        (shipfail/crew)               (shipfail/thoth)

  boot protocol + syscalls     agents + skills + tools       code + worktree
  (how to boot)                (userland / $HOME)            (the app)

            ▲                          ▲                          ▲
            |                          |                          |
            └──────────────── read AGENTS.md boot file ────────────┘
```

In normal Linux terms:

- `promptware` ≈ kernel + early boot rules
- `crew` ≈ distribution userland (`/home`, packages, policies)
- `thoth` ≈ the application you’re working on

In PromptWareOS terms:

- **`AGENTS.md` is your bootloader**.
- It mounts:
    - the local repo as `/work`

    - Crew as `$HOME`
- Then it selects an init agent.

---

6) From zero: “convention-only boot” in a single file

This is the point: you don’t need a giant framework or a plugin to start.

To boot PromptWareOS in your repo, you only need **one canonical file**:

> ***AGENTS.md***

And then you keep other tool entrypoints as thin shims.

6.1 The canonical boot file: `AGENTS.md`

Put this at the root of your project repo (example: `shipfail/thoth`).

```
# PromptWareOS Boot

BOOTLOADER:
- Load PromptWareOS boot protocol: https://github.com/shipfail/promptware
- Mount the project worktree as: /work
- Mount Ship.Fail Crew as HOME: os://crew

MOUNT:
- os://crew = https://github.com/shipfail/crew@v0.1.0

INIT:
- Activate agent: os://crew/agents/cartographer/agent.md
- Apply profile overlay (optional): os://crew/profiles/thoth/profile.md
```

Read it like Linux:

- `MOUNT` is your `/etc/fstab`.
- `os://crew` is a logical mount point.
- The GitHub URL is the physical location.

Once mounted, the agent should refer to *logical* paths:

- `os://crew/agents/...`
- `os://crew/skills/...`

Not raw GitHub URLs.

That’s how we keep promptware **portable**.

6.2 Thin shims: `CLAUDE.md`, `GEMINI.md`, `CODEX.md`

Most coding agents look for their own entrypoint file. Don’t duplicate the bootloader.

Make them forward to `AGENTS.md`.

Example `CLAUDE.md`:

```
This repo boots via **AGENTS.md**.

Please read `AGENTS.md` and treat it as the canonical PromptWareOS bootloader.
```

Do the same for `GEMINI.md` / `CODEX.md`.

One source of truth.

---

7) Determinism vs rolling release (Debian vs Arch, on purpose)

A distribution lives and dies by upgrade discipline.

Crew supports two modes.

7.1 Pinned (default): deterministic like Debian stable

Pin Crew to a tag or commit.

- predictable behavior
- reviewable upgrades
- reproducible runs

Example:

- `https://github.com/shipfail/crew@v0.1.0`
- `https://github.com/shipfail/crew@<commit-sha>`

7.2 Floating (optional): rolling release like Arch

Mount `@main` if you want the newest improvements immediately.

- fast iteration
- sometimes breaking changes
- great for experiments

Example:

- `https://github.com/shipfail/crew@main`

Rule of thumb:

- **Teams ship with pins.**
- **Solo hacking can float.**

---

8) Why this matters (even if you never say “PromptWareOS” out loud)

The three-repo model buys you leverage.

Upgrade once, reuse everywhere

Improve an agent. Tighten a skill. Clarify a tool contract.

Every repo that mounts Crew benefits—without copy/paste.

Keep project repos clean

Your project repo stays a normal repo.

- code
- tests
- docs
- and one small `AGENTS.md` bootloader

No prompt bloat.

Turn agents into portable artifacts

An agent becomes something you can point to:

- versioned
- documented
- mountable
- sharable

That’s how we get from “prompt folklore” to “prompt infrastructure.”

Today: one agent. Tomorrow: a crew.

MVP boots one init agent.

Later, the same distro model supports:

- multiple personas
- teams
- role routing
- skill packs

Linux didn’t start with containers. It started with a bootloader and a userland.

---

9) Fork Crew and build your own distro

Here’s the call to action:

Step 1 — Fork the distro

Fork `shipfail/crew` and make it yours.

- Add your own agents
- Add your own skills
- Add your own tool contracts

Step 2 — Boot it in a repo

In any repo, add `AGENTS.md` and mount your distro.

```
# in AGENTS.md (conceptually)
os://crew = https://github.com/<you>/crew@v0.1.0
```

Step 3 — Ship upgrades like a distro maintainer

- Tag releases
- Write changelogs
- Pin in production repos
- Float only when you mean to

Because in the AI era, your repo doesn’t just build code.

> ***It boots intelligence.***

Links

- PromptWareOS kernel: https://github.com/shipfail/promptware
- Ship.Fail Crew distro: https://github.com/shipfail/crew
- Example app repo: https://github.com/shipfail/thoth
- Background reading:
  - Booting Intelligence (PromptWareOS v0.2): https://ship.fail/blog/2025/12/18/booting-intelligence-architecture-promptware-os-v0-2/
  - PromptWareOS & Prompt-Driven Architecture (PDA): https://ship.fail/blog/2025/12/17/promptware-os-prompt-driven-architecture-pda/

promptware promptwareos pda agents linux distribution shipfail

Written by Huan Li Follow

Founding Human & Architect | Author @ Wechaty, Silicon Adventist, Serial Entrepreneur, Angel Investor, Burner🔥

Loading comments…