

■ // DEEPSEEK HARNESS · SETUP

Ein Befehl, fünf Profile — und der Sicherheitshinweis

DeepSeek Harness startet mit einem einzigen npx-Befehl und öffnet eine Web-Oberfläche auf deinem Rechner. Darunter liegt eine Architektur, in der wirklich jeder Teil ein Plugin ist — auch der Agent Loop. Dieser Guide zeigt den ersten Start, wie du die Zusammensetzung siehst und veränderst, und was DeepSeek selbst über die Risiken schreibt.

MIT-LIZENZ

EIN BEFEHL

5 PROFILE

DEVELOPER PREVIEW

QUELLEN INKLUSIVE

Worum es geht: Harness heißt im Terminal `dsh` und liegt unter der MIT-Lizenz. Das Besondere steht in der Architektur-Doku wörtlich: `Modelladapter, Tool-Registry, Session-Log und der Agent Loop selbst sind Plugins` — jedes davon aus der Konfiguration heraus austauschbar. Und der Satz, der die ganze Idee zusammenhält: „There is no privileged core to patch.“ Du erweiterst nichts, indem du einen Kern anpasst, sondern indem du ein Plugin daneben hängst.

1 Der erste Start

Node.js installieren, dann ein Befehl:

```
npx @deepseek-ai/dsh web
```

Das startet die Web-Oberfläche auf `http://127.0.0.1:3080` und öffnet deinen Browser. Mit `--no-open` läuft der Server, ohne dass ein Fenster aufgeht — praktisch, wenn du über SSH arbeitest, denn dort wird nur die Host-URL ausgegeben.

Aus dem Quellcode

```
git clone https://github.com/deepseek-ai/deepseek-harness.git
cd deepseek-harness
pnpm install
pnpm run build
pnpm dsh web
```

`pnpm run build` erzeugt die Artefakte, `pnpm dsh web` nutzt sie danach ohne neuen Build. Achtung: Der Standard-Branch heißt `master`, nicht `main`.

2 Profile — fünf fertige Zusammensetzungen

Ein laufendes `dsh` ist ein Plugin-Baum, der beim Start aus geordneten Schichten zusammengesetzt wird. Welche Schichten das sind, bestimmt das Profil:

```
// mitgelieferte Profile
web          Browser-Anwendung (Standard, live nachladbar)
headless     Ein-Durchlauf-Runner, ohne Server
sdk          JSON-RPC-Server für das SDK
sdk-minimal  eigener, vollständig explizite SDK-Baum
acp          Automatisierung, ACP-Server
```

Die gemeinsame Grundsicht heißt `dsh-base` und bringt Modelladapter, Tools, Persistenz, Sandbox- und Freigabe-Policy, Einstellungen, Zugangsdaten und Telemetrie mit. Darauf setzt je ein Bundle für die Anwendung auf, etwa `dsh-web-app`.

Nur das `web`-Profil lädt Änderungen im Betrieb neu. `headless`, `sdk` und `acp` wenden alle Schichten einmal beim Start an — bei einem Ein-Durchlauf-Programm würde ein Austausch mitten in der Arbeit den Lebenszyklus zerreißen.

3 Sehen, was dein Rechner startet — und es ändern

Der wichtigste Befehl für den Einstieg, weil er den kompletten Baum ausgibt:

```
dsh --profile web --dump-config
```

Jede Zeile, die dort erscheint, kannst du durch einen eigenen Patch ersetzen. Ein Patch greift eine Zeile über ihre `id` und tauscht deren komplette Konfiguration aus, oder fügt neue Zeilen ein. Die Schichten werden in dieser Reihenfolge angewendet:

```
// von unten nach oben, spätere gewinnen
1 die Bundles des Profils, in ihrer Reihenfolge
2 cordis.patch.yml des Profils
3 cordis.patch.yml im Harness-Home
4 Overlay aus --patch
```

Externe Plugins installierst du dauerhaft über `dsh plugin`. Wer eigene Plugins veröffentlicht, setzt das GitHub-Topic `dsh-plugin` — darüber findet die Community sie.

4 Warum „alles ist ein Plugin" mehr ist als ein Slogan

Unter `dsh` liegt **Cordis**, ein Framework, in dem Plugins Dienste, typisierte Events und *umkehrbare* Effekte in einen gemeinsamen Kontext einbringen. Der zweite Teil ist der entscheidende: Registrierungen sind Effekte, die zurückgerollt werden, sobald ihr Plugin entladen wird. Deshalb lässt sich der Baum im laufenden Betrieb umbauen, ohne Reste zu hinterlassen.

Praktisch heißt das: Du willst einen anderen Modelladapter, ein eigenes Tool-Set, ein anderes Session-Log oder einen eigenen Agent Loop? Alle vier sind Zeilen in derselben Konfiguration. Es gibt keine Stelle, an der du das Framework selbst anpassen müsstest.

5 Der Sicherheitshinweis — bitte vorher lesen

DeepSeek legt dem Projekt eine eigene `SAFETY.md` bei, und die ist deutlich. Es steht dort wörtlich, dass die Software **experimentell** ist, **kein Sicherheitsaudit** hinter sich hat und nicht als produktionsreif behandelt werden darf. Im README dazu in Großbuchstaben: es *wird* Änderungen geben, die die Kompatibilität brechen.

Der Grund für die Warnung liegt in der Natur der Sache: Das Projekt führt vom Modell erzeugten Code und Befehle aus, lädt Plugins von Dritten und greift auf Netzwerk, Prozesse, Zugangsdaten und Dateien zu, die du ihm zugänglich machst. Sandbox und Freigabe-Abfragen senken das Risiko, garantieren laut Doku aber keine Isolation — und schützen ohnehin nichts, worauf das Projekt zugreifen darf.

Die Empfehlungen aus der Doku

- Mit den **geringsten nötigen Rechten** laufen lassen
- Am besten in einer **wegwerfbaren VM**, einem Container oder einer eigenen Umgebung
- **Backups** von allem, worauf das Projekt zugreifen kann
- Plugins, Konfiguration und vorgeschlagene Befehle **vorher ansehen**
- Nicht als einzige Sicherheitsmaßnahme für fremde Workloads verwenden

6 Quellen

- **Repo (MIT):** github.com/deepseek-ai/deepseek-harness
- **Dokumentation:** deepseek-harness.github.io/deepseek-harness/
- **Architektur:** `docs/architecture.md` im Repo
- **Sicherheitshinweis:** `SAFETY.md` im Repo
- **Cordis (das Framework darunter):** github.com/cordiverse/cordis
- **Plugins der Community:** GitHub-Topic `dsh-plugin`

■ // AUF EINEN BLICK

Die Kurzfassung

Vier Punkte für den Start.

- ✓ `npx @deepseek-ai/dsh web` startet die Oberfläche auf **Port 3080**
- ✓ `--dump-config` zeigt den Plugin-Baum — **jede Zeile ist ersetzbar**
- ✓ Auch **Agent Loop und Modelladapter** sind Plugins, kein privilegierter Kern
- ✓ Developer Preview **ohne Sicherheitsaudit** — in einer VM oder einem Container testen

■ // BLEIB DRAN

Agenten-Tools, mit dem Kleingedruckten.

Ich lese bei solchen Releases auch die SAFETY.md, bevor ich etwas empfehle. Folg mir für die nächste Einordnung.

@ki.borish

Fakten verifiziert am 01.09.2026 aus README.md, docs/architecture.md und SAFETY.md im Repository github.com/deepseek-ai/deepseek-harness (MIT-Lizenz, Stand 207.720 Sterne) sowie der GitHub-API. Das Projekt ist Developer Preview und ändert sich schnell — Befehle und Profilnamen können mit neuen Releases abweichen. Prüfe die Doku im Repo gegen deine Version.