WHITE PAPER · SEPTEMBER 2026 · VERSION 1.0

Agent Hub

A cloud workspace for AI agents. One link, one workspace per person, nothing to install. Built entirely from open-source components on top of JupyterHub, so an institution that already runs JupyterHub can host it as it stands.

Fengyuan Zhu
Research Assistant Professor
Siebel School of Computing and Data Science
University of Illinois Urbana-Champaign

Web fengyuan-zhu.com/agent-hub
Demo film vimeo.com/1226732315 (7 min, recorded live on the running deployment)
Contact zhufy@illinois.edu

> **What it is.** An agent such as Claude Code or opencode can do far more than a chat window, because it has a real computer and knows how to use it: it reads and writes files, runs code, installs what it needs, looks things up, and keeps going until the job is done. The hard part in practice is getting one running on everyone's own machine. Agent Hub puts the whole thing in the cloud. A person opens a link, signs in, and is in their own workspace with an agent that is ready to work. Their files stay there between visits, and a job keeps running after the browser tab is closed.
>
> **What it is made of.** JupyterHub does sign-in and starts one container per person. Inside the container run Pi, an open-source agent engine, and Pi Web, its browser interface. Every model call and every web lookup goes through a small gateway that holds all credentials and writes one ledger. No provider key ever enters a container. Everything below the gateway is interchangeable: any OpenAI-compatible model endpoint, whether an institutional gateway, a commercial subscription or a server on your own hardware.

1. The problem it solves

Chat assistants reached ordinary people because they are a URL. Agents have not, because an agent needs a machine to act on, and the usual way to give it one is to install a command-line tool on the user's own computer, register an API key, learn a terminal, and work through a provider's sign-in flow. Each of those steps loses people, and all of them sit in front of the first useful result. For most staff, students and researchers the answer to "would you like to try an agent?" is decided by that installation, not by what the agent could do.

Agent Hub removes the installation by moving the computer rather than the person. The agent's machine is a container in the cloud, opened with a browser. Nothing is configured on the user's side. What the person sees is the agent and their files, not a development environment with an agent hidden inside it.

2. What a person gets

A link and a sign-in. The first screen is a sign-in page, not a product tour. After it, a panel lists what the agent can do and offers three example requests that can be sent with one click, so nobody has to invent the first sentence.

A workspace of their own. Each person has a private container with a home folder. Files uploaded from their machine, files the agent writes, charts it draws and pages it builds all live there and can be opened, edited or downloaded from the browser. An HTML file the agent produces renders in place, and the source is one click away.

An agent that acts. The agent reads and writes files, runs shell commands, installs the packages a task needs, fetches web pages, searches the web, delegates side questions to a subagent, keeps a long-term memory across conversations, and plans multi-step work. These abilities are plugins; five come switched on, any can be switched off, and more can be added from the settings panel or by asking the agent for one.

Work that outlives the tab. The job runs in the container, not in the browser page. In the demo film the tab was closed in the middle of a long task; the agent carried on for another 1 minute 16 seconds with nobody watching and the finished files were waiting when the page was reopened. A long job can be left alone.

A choice of models. The settings panel lists the models the deployment provides. A person may also paste a key of their own for a provider they have access to; the models it unlocks appear in the same list, and their usage is charged to that key rather than to the shared pool.

3. Three things it did in the demo

All three were recorded on the running deployment, in one take each, with no rehearsal or replay. The numbers below were measured from the recordings.

CASE ONE

A messy spreadsheet

`read · bash · the model · write`

A workshop sign-up sheet uploaded from the organiser's laptop: 53 rows, three date formats, e-mail addresses in mixed case, "year" written six ways, and a few people who signed up twice. The agent read the file, reported what was wrong, installed the library it needed, cleaned the data to 50 people, read all fifty free-text answers and grouped them by meaning, drew the charts and wrote a self-contained HTML report that downloads in one press.

CASE TWO

Checking a number

`webfetch · subagent + websearch · write · memory`

Whether a monthly jobs figure had been revised after publication. The agent fetched the page and picked out the numbers, then sent a subagent to verify the figure independently against a second source. It wrote the result up with both values and a link to every source, and saved the conclusion to long-term memory. In a brand-new conversation, with none of that work present, the agent still answered from memory.

CASE THREE

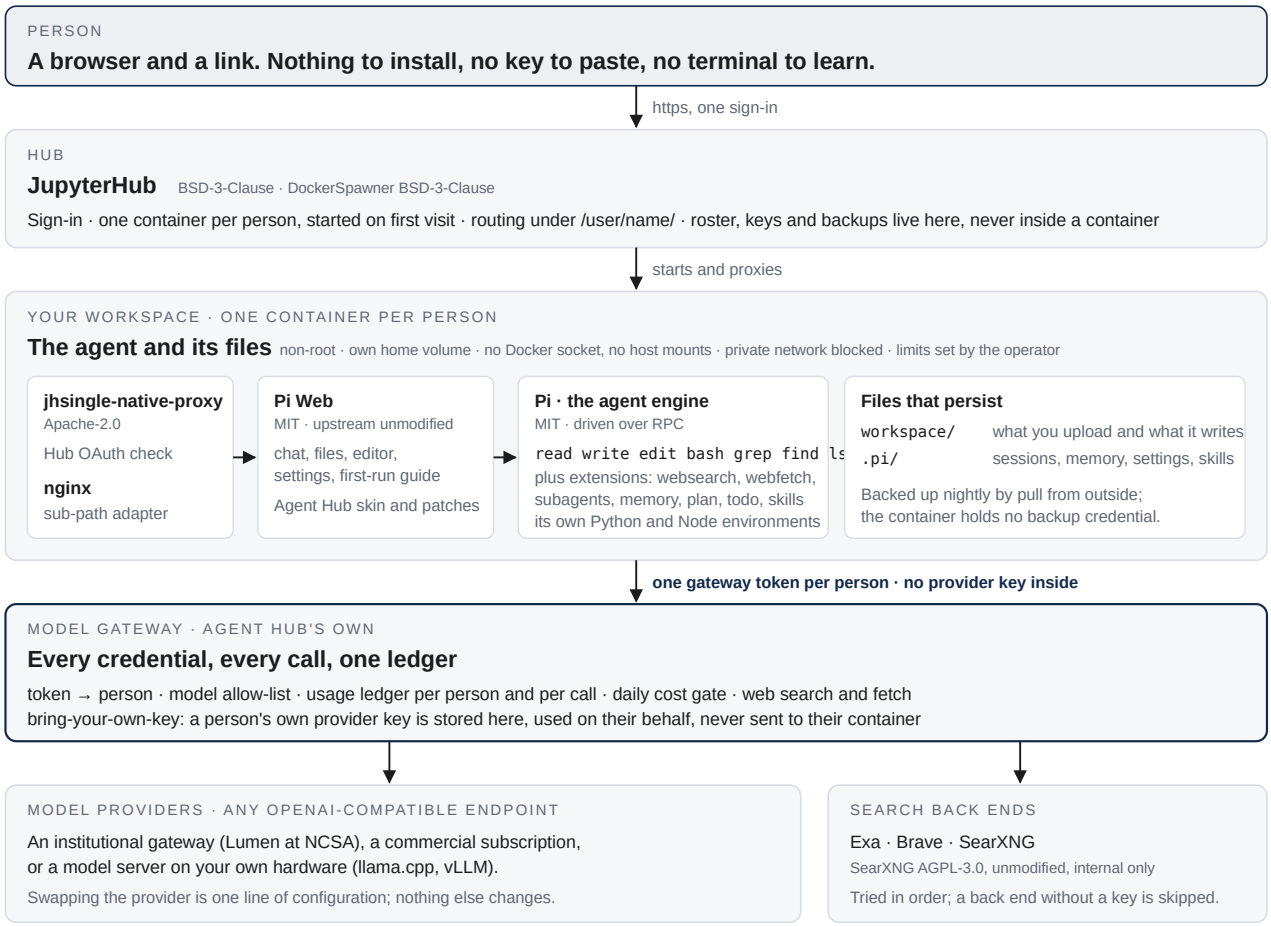
A page that already exists

`read · edit · bash`

A sign-up browser page written in an earlier session, opened and used in the workspace. Looking one person up showed them twice: the page counted rows, not people. Told what was wrong, the agent read the page, worked out which rows were the same person, and edited the existing project. The count went from 53 rows to 50 people.

Three things carry across all of them: it works on the files you already have, it comes with tools for the web and a memory and more can be added, and the files persist so you can come back to them.

4. How it is built



The layers, top to bottom. The Hub and the gateway are the only places credentials live. A container holds one gateway token that identifies its owner and nothing else.

Design rules

No provider key enters a container.
The gateway holds the model keys and the search keys. A container is given one token that identifies its owner. Tests log into a real container and check that no other credential is present.

One ledger for everything.
Every model call, every search and every fetch, including the refused ones, is written to a server-side ledger keyed to the person. Per-person and per-session token counts, tool calls, duration and cost are reconciled against the agent's own session files and must match.

One container per person, isolated.
Non-root, its own home volume, no Docker socket, no host mounts, no reach into the private network, and CPU, memory and process limits set by the operator. A person can only open their own workspace; another person's path is refused.

The Hub is JupyterHub.
Sign-in, per-user containers and routing are JupyterHub doing what it already does on campus clusters. The notebook interface is removed; the person sees the agent. Institutional sign-in is a change of authenticator, not a rewrite.

The provider is a setting.
The gateway speaks the OpenAI chat-completions protocol upstream. The same deployment has run on an institutional gateway (Lumen at NCSA), on a commercial subscription and on a local model server. Model context windows, output limits and modalities are read from the provider, not typed in.

Backups pull, never push.
A nightly job outside the platform copies each person's workspace and agent state into dated snapshots. The containers hold no backup credential, and the receiving side never deletes.

Verified, not assumed

A fourteen-part test suite runs against the live deployment. It covers sign-in and cross-user isolation, the sub-path proxying that lets the interface run under one Hub, a full tool-calling loop against a real model, streaming, work continuing after the browser is gone, many abandoned streams not blocking a workspace, the first-run guide, the settings panel writing real settings into the container, bring-your-own-key never being readable back, the file browser and editor writing to disk, skills and pasted images reaching the model, web search and fetch going through the gateway and landing in the ledger, private-network egress being blocked, and the account open-and-close path. Every check reads its evidence from the result side, on disk or in the ledger, not from what the screen shows.

5. The open-source chain

Agent Hub is a thin layer of configuration and glue on top of published open-source projects. The upstream projects are used as released; where the interface is adjusted, it is done by patches applied at start-up, not by forking.

COMPONENT	ROLE IN AGENT HUB	LICENCE	NOTE
JupyterHub 5.4	Sign-in, one container per person, routing, admin	BSD-3-Clause	Already run by many campuses and clusters
DockerSpawner 14	Starts each person's container with the right limits and volume	BSD-3-Clause	Swappable for a cluster spawner

COMPONENT	ROLE IN AGENT HUB	LICENCE	NOTE
jhsingle-native-proxy 0.8	Lets a non-notebook web app sit behind the Hub's OAuth	Apache-2.0	Upstream archived; small and stable
nginx	Sub-path adaptation inside the container	BSD-2-Clause	
Pi 0.85	The agent engine: tools, sessions, extensions, driven over RPC	MIT	No SaaS back end; any OpenAI-compatible provider
Pi Web 1.21	The browser interface: chat, files, editor, settings	MIT	Upstream unmodified; skin and patches applied at start-up
Pi extensions and skills	Subagents, memory, plan mode, task list, YouTube transcripts	MIT	Published by the Pi authors; web search and fetch are Agent Hub's own
SearXNG	Last-resort search back end behind the gateway	AGPL-3.0	Unmodified, internal network only; optional
Docker, Python, Node.js, uv	Containers and the runtime the agent programs in	Apache-2.0 / PSF / MIT	

Agent Hub's own code is the Hub configuration and page templates, the model gateway, the container entry point, the interface skin and patches, a few Pi extensions, the account tooling, the backup job and the test suite. The AGPL component is the only copyleft item in the chain; it is neither modified nor exposed, and a deployment that configures commercial search back ends only has no copyleft component at all.

6. What a deployment needs

- A host that runs Docker, or an existing JupyterHub with a spawner. A single virtual machine has been enough for a class-sized group.
- An OpenAI-compatible model endpoint and one key for it, held by the gateway. An institutional gateway is the natural choice; a subscription or a local model server also works.
- Optionally, keys for one or more search providers. A back end without a key is skipped.
- A domain name with TLS in front. The reference deployment terminates TLS on a small edge machine and reaches the Hub over an outbound tunnel, so nothing inbound is opened at the host.

Operating it is three commands: one pushes configuration and rebuilds images, one runs the full test suite against the live site, and one opens or closes an account. Opening an account prints the link and a one-time password; closing it revokes the gateway token and stops the container while keeping the person's files. Neither restarts anything for anyone else.

Per-container resource limits, disk and cost gates are settings for the operator; they are not properties of the product and are not stated here.

7. What it is not yet

- **Not an institutional service.** The reference deployment uses its own roster and per-person passwords. Institutional single sign-on is a supported change of authenticator but has not been exercised with a real identity provider.
- **The gateway is a front door, not a wall.** Web search and fetch go through it and are accounted for. A shell command can still reach the public internet directly. Private-network egress is blocked; a full outbound policy is future work.
- **Trust has a cost.** A project the agent clones may contain extensions that run when it is opened, with the agent's own permissions. The container is the boundary; the first-run guide says this plainly.
- **No disk quota.** CPU, memory and process counts are limited; disk is not. Nightly backups exist, but the guide tells people to keep their own copy of anything important.
- **Third-party interface.** Pi Web is an upstream project, and the proxy component's repository is archived. Both are small and have been stable, but long-term maintenance is a responsibility to assign, not to assume.
- **Connected data is next, not now.** Giving the agent a person's institutional mailbox or file store is a matter of gateway extensions and policy approval, and an agent that both browses the web and holds a mailbox token needs a confirmation step on every write. Neither is built.

8. Status

Agent Hub runs today at agent-hub.fengyuan-zhu.com, by invitation. It has been used by test accounts, by a small number of invited people on real work, and for the recorded demo. The next step is a pilot with people who are not its author, in a class or a research group, using an institutional model gateway and institutional sign-in. Whether people use an agent when the only thing between them and it is a link is the question this was built to answer.

Agent Hub is a research prototype by the author and is not an official service of the University of Illinois. Component names and licences are those of the respective upstream projects. This document describes the deployment as of September 2026.