DevOps for research

Jun Kato

AIST, Arch Inc.

#non_research_tips #情報科学研究補助技法

Links to open-source code in this lecture

- `arcatdmz/non-research-tips-gitlab-ci`
  - GitLab CI/CD simple example
  - https://gitlab.com/arcatdmz/non-research-tips-gitlab-ci
- `arcatdmz/non-research-tips-devcontainer`
  - VS Code DevContainer for consistent development experience
  - Node.js + MySQL + Prisma
  - https://github.com/arcatdmz/non-research-tips-devcontainer
- `arcatdmz/acm2024-primary-article-template`
  - Editing, linting, and sharing papers
  - Overleaf + GitHub Actions + GitHub Pages + Slack
  - https://github.com/arcatdmz/acm2024-primary-article-template
- `arcatdmz/arcatdmz.github.io`
  - Single source of truth for multiple views
  - Gulp + Pug + BibTeX etc.
  - https://github.com/arcatdmz/arcatdmz.github.io
- `arcatdmz/research-assistant-playbook`
  - Agentic development of agentic research assistant
  - https://github.com/arcatdz/research-assistant-playbook

DevOps for research: agenda

- Development environment setup
- Continuous integration/delivery
- Web technologies for development/public relations
- Agentic development of your agentic assistant

Development environment setup

DevOps for research

Development environment setup

- Package managers for Windows/macOS
- Two-factor authentication for improved security
- SSH config for custom networking
- DevContainer for consistent development experience

Package managers for Windows/macOS

- Windows
  - winget [**recommended**]
  - scoop, Chocolatey
- macOS
  - Homebrew [**recommended**]
  - macPorts

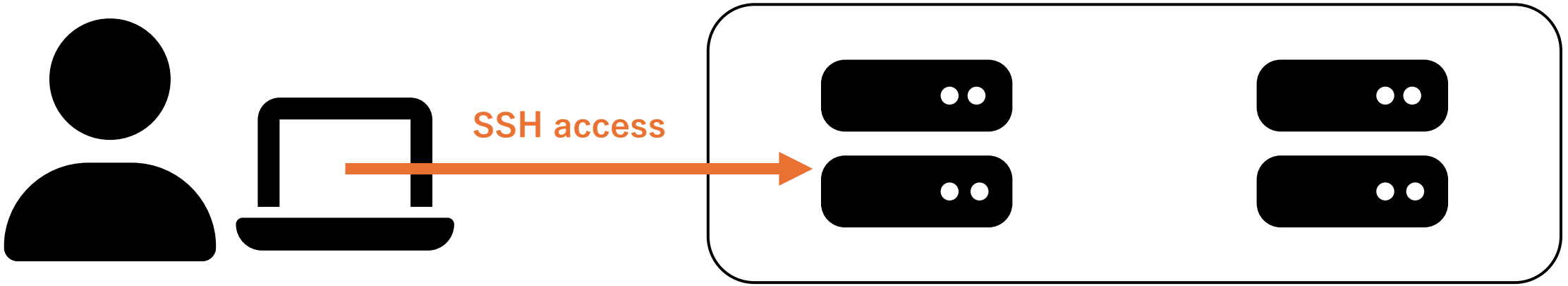
Two-factor authentication for improved security

- Microsoft Authenticator
- Authy [**recommended**]
- 1Password

SSH config

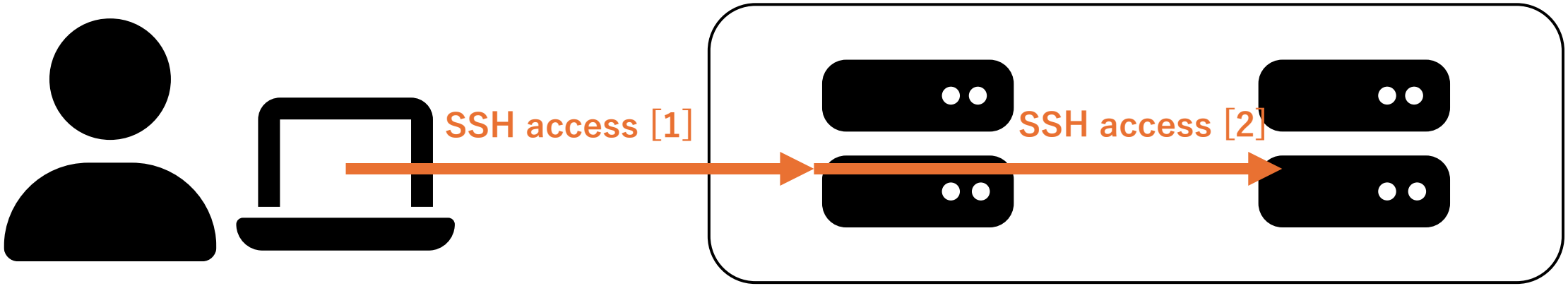
- Cf. Week 1: SSH + server coding
- You can do a lot with SSH, probably more than you'd expect

SSH config case studies 1:
`ProxyJump` for jumping hosts



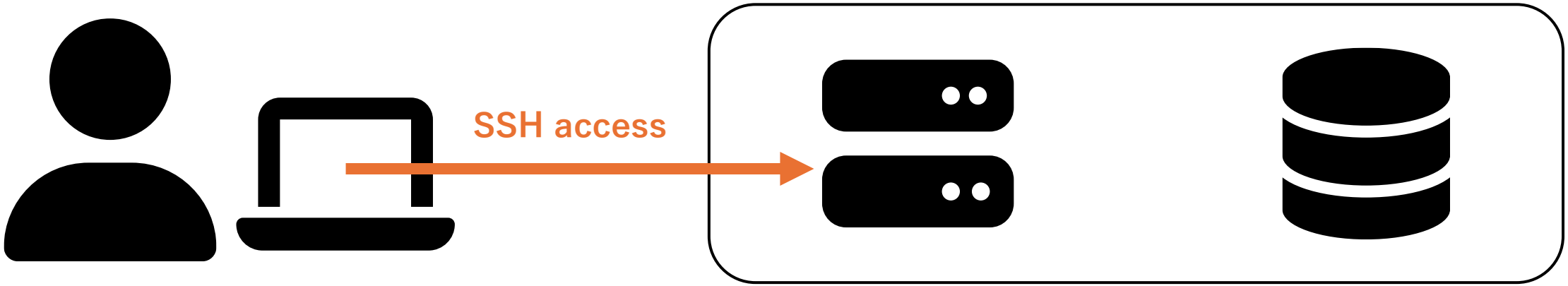
- Suppose you have a server that can only be accessed from within a restricted network that includes another server
- You have SSH keys to access both the bastion host (intermediate server) and the final destination server

SSH config case studies 1:
`ProxyJump` for jumping hosts



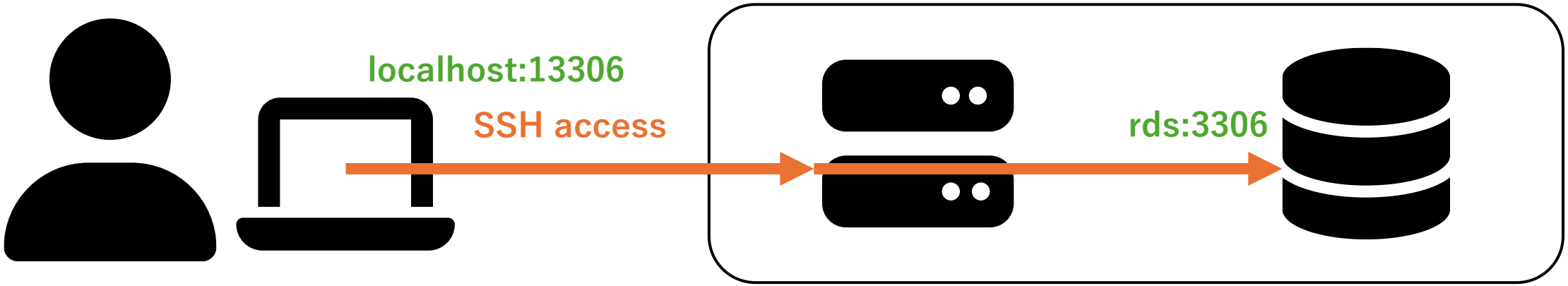

```
1 Host ssh-config-proxy-jump
2     HostName xxx.xxx.xxx.xxx
3     ProxyJump ssh-config-bastion
4 Host ssh-config-bastion
5     HostName final-destination
```

SSH config case studies 2:
`LocalForward` for remote service access



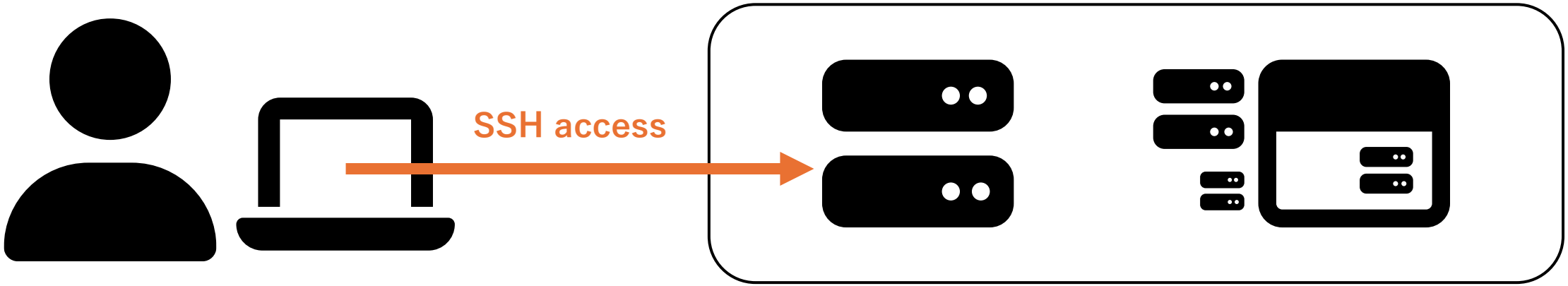
- Suppose you have a database server that can only be accessed from within a restricted network that includes an app server
- You have SSH access to the app server, but not to the database server

SSH config case studies 2:
`LocalForward` for remote service access



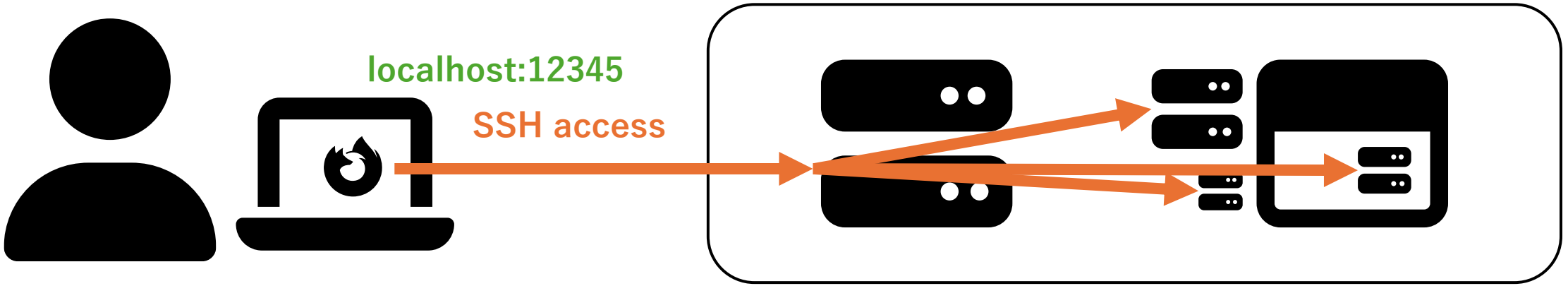

```
1 Host ssh-config-localforward
2     User ec2-user
3     HostName xxx.xxx.xxx.xxx
4     IdentityFile ~/.ssh/xxx.pem
5     LocalForward 13306 rds:3306
```

SSH config case studies 3:
`DynamicForward` for SOCKS v5 proxy



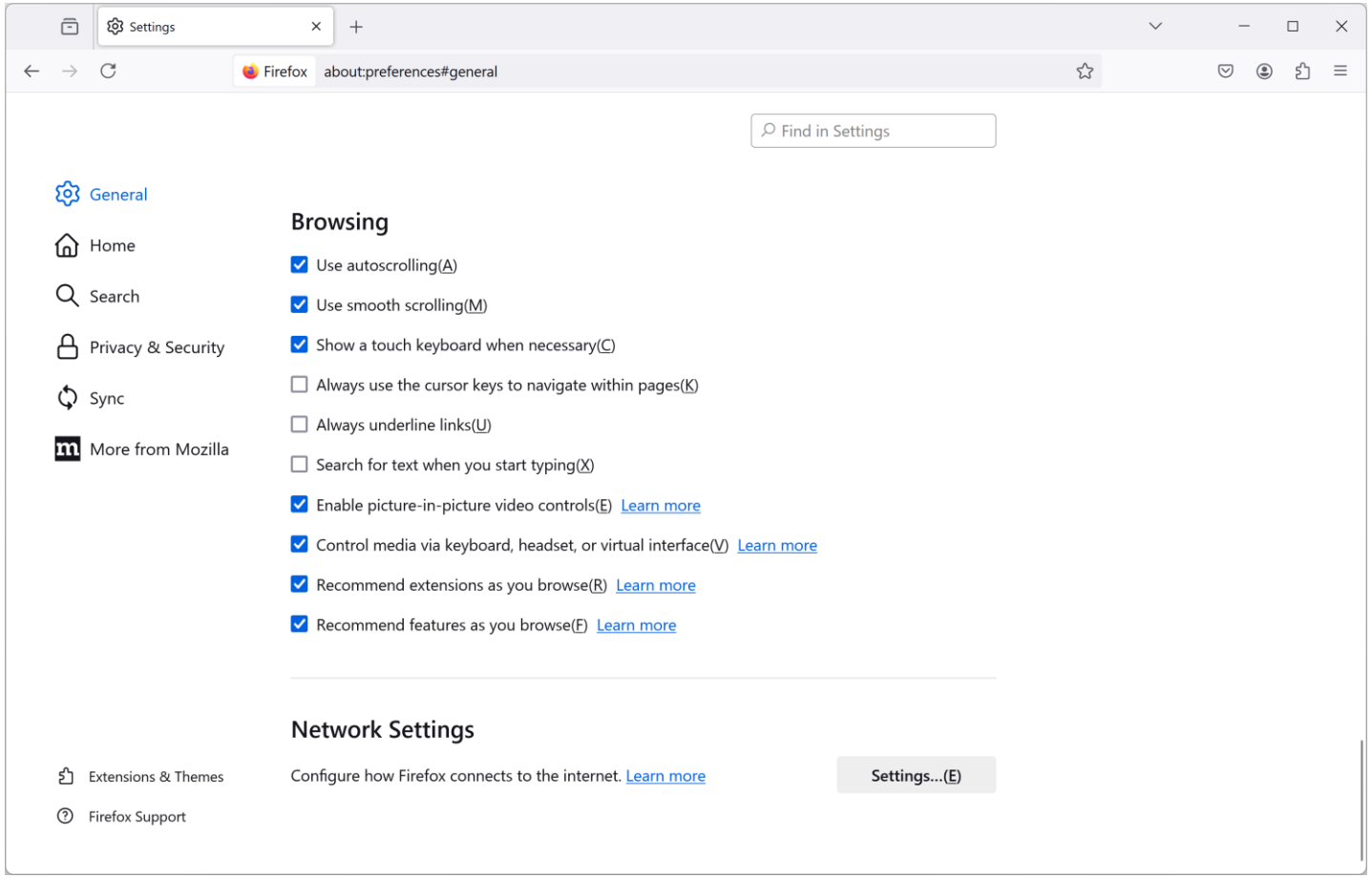
- Suppose you have a service that can only be accessed from within a restricted network that includes an app server
- You have SSH access to the app server, but the service runs on multiple servers to be accessed via a web browser

SSH config case studies 3:
`DynamicForward` for SOCKS v5 proxy




```
1 Host ssh-config-dynamicforward
2     User remote-user
3     HostName xxx.xxx.xxx.xxx
4     IdentityFile ~/.ssh/xxx.pem
5     DynamicForward 12345
```

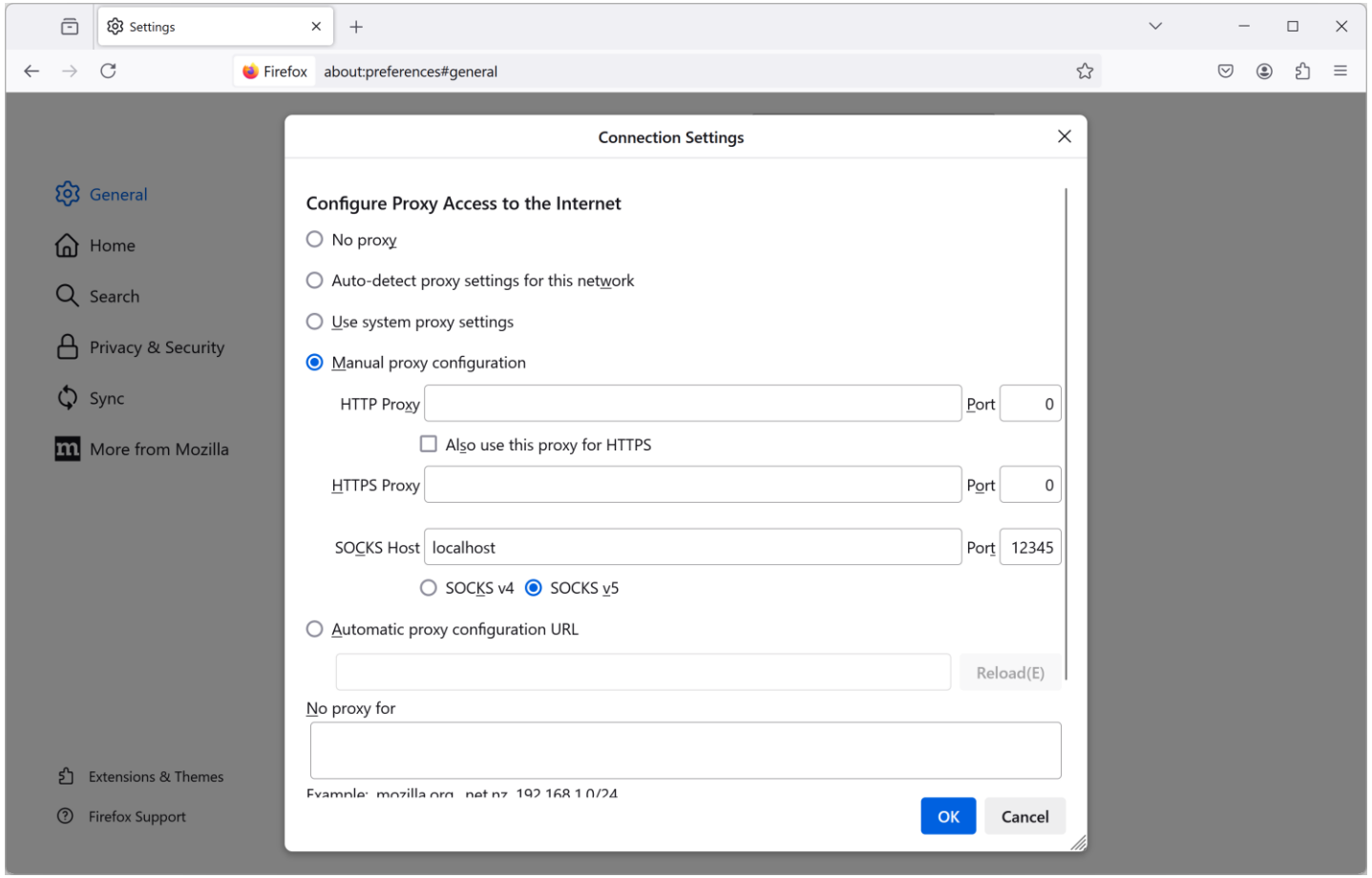
SSH config case studies 2:
`DynamicForward` for SOCKS v5 proxy



Firefox browser config:

Network Settings

SSH config case studies 2:
`DynamicForward` for SOCKS v5 proxy



Firefox browser config:

Network Settings > Manual proxy configuration >

SOCKS Host, Port, SOCKS v5

DevContainer (VS Code feature)

- Prerequisite: Docker Desktop (or Rancher Desktop)
- Useful feature for Windows: WSL2
- DevContainer for consistent development experience

DevContainer (VS Code feature): Prerequisite

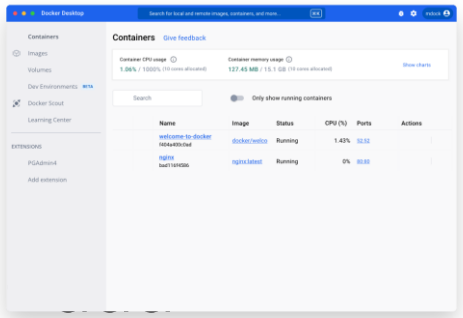
Docker Desktop

- Free for educational, non-commercial, personal, and small business use

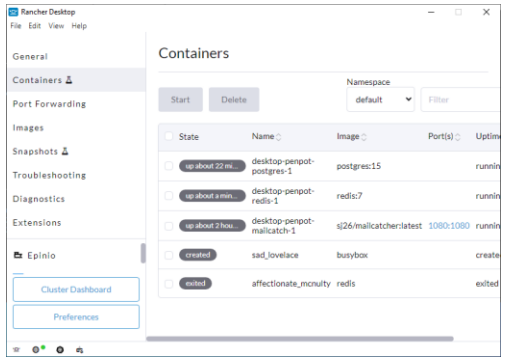
Rancher Desktop

- Completely OSS, free for anyone who agrees the terms of use

- **`docker`** and **`docker-compose`** command support



https://www.docker.com/wp-content/uploads/2023/08/docker-desktop-hero-v2.svg



https://suse-rancher-media.s3.amazonaws.com/desktop/v1.14/ui-main/Windows_Containers.png

DevContainer (VS Code feature): Useful Windows feature: WSL2

- Windows Subsystem for Linux 2
- To install and start using bash on Ubuntu:
  - **`wsl --install`** in the administrator privileged console
  - Restart system
  - **`wsl --set-default-version 2`** in the admin console
  - **`bash`** in console

```
PS C:\Users\junkato> bash
arc@laptop:/mnt/c/Users/junkato$ ls | grep Desktop
drwxrwxrwx 1 arc arc      512 Jun 22 01:04 Desktop
```

DevContainer (VS Code feature) for consistent development experience

- Use the same runtime (e.g., Node.js)
- Use the same package manager (e.g., npm)
- Use the same backend server (e.g., MySQL)
- Use the same VS Code extensions, editor config, etc.

…

Want a consistent development experience across machines?

DevContainer has you covered!

DevContainer (VS Code feature) for consistent development experience

```jsonc
{
  "name": "Example dev container",
  "dockerComposeFile": ["./docker-compose.yml"],
  "service": "node",
  "workspaceFolder": "/workspace",
  "customizations": { "vscode": {
    "extensions": [
      "esbenp.prettier-vscode",
      "EditorConfig.EditorConfig",
    ]
  } },
  "forwardPorts": [3000, "mysql:3306", "prisma:5555"],
}
```

Example devcontainer.json

DevContainer (VS Code feature) for consistent development experience

```yaml
version: "3.7"
services:
  mysql:
    image: mysql:8
    healthcheck: { test: ["CMD", "mysqladmin ping -h localhost"] }
  prisma:
    # mysql client
  node:
    build:
      context: ../
      dockerfile: ./.devcontainer/Dockerfile
    command: sleep infinity
    # more config entries omitted
```

Example docker-compose.yml

By the way...
Share beautiful images of your code

```
# What does this do?
## Carbon lets you create and share beautiful images of your source

You know all of those code screenshots you see on Twitter? Although
thought there was room for improvement in the aesthetic department.
try it out and impress all your developer and designer friends. 🎨

## Import
There are a few different ways to import code into Carbon:

- Drop a file into the editor
- Append a GitHub gist id to the url (example)
...
```

https://carbon.now.sh/

Continuous integration/ continuous delivery

DevOps for research

Continuous integration/continuous delivery

- GitHub or GitLab?
- Self-hosted runners
- CI/CD use cases 1: linting papers
- CI/CD use cases 2: running heavy workloads on clouds
- CI/CD use cases 3: ChatGPT integration
- CI/CD use cases 4: publishing websites

GitHub or GitLab?
from CI/CD perspective

GitHub Actions

- Not much difference for basic usage
- writes config under `~/.github/workflows/`
- **Pros:** Has a vast library of community-defined "Actions" useful for various purposes
- **Cons:** You're vendor-locked!

GitLab CI/CD

- writes config in `~/.gitlab-ci.yml`
- **Pros:** Can be hosted entirely on your own servers without worrying much about resource limits
- **Cons:** Maintenance costs

GitLab CI/CD basic example

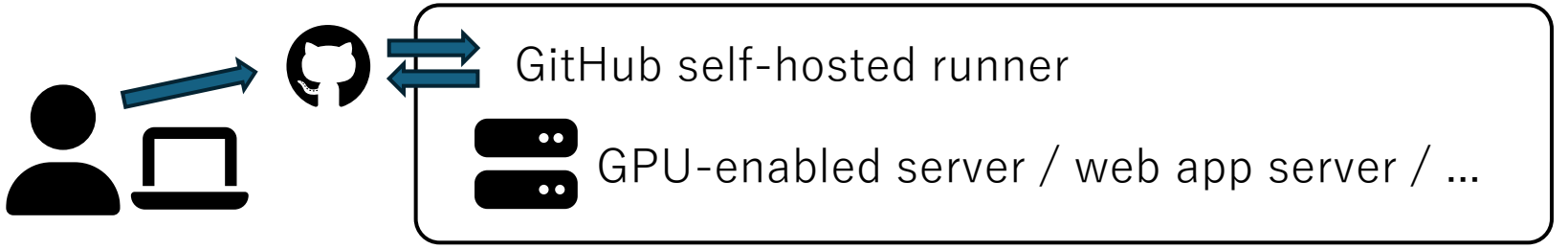
.gitlab-ci.yml

```yaml
image: docker:dind

before_script:
  - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_R

stages:
  - build
  - deploy

variables:
  IMAGE_PATH: ${CI_REGISTRY}/${CI_PROJECT_NAMESPACE}/${CI_PROJECT_NA
  CONTAINER_NAME: ${CI_PROJECT_NAME}-${CI_JOB_NAME}
# more config entries omitted
```

Example .gitlab-ci.yml

Self-hosted runners
Basics and usages

- GitHub Actions and GitLab CI/CD both support self-hosted runners – CI/CD workers that run on machines under your control, such as **modern NAS devices** and various servers
- These are especially useful if
  - you have **a handy machine** (**or some special machines**) and want to avoid being charged for running an automated process
  - you have **a deployment target on cloud computing platforms** and want to securely update running services on each push



CI/CD case studies 1:
Editing, linting, and sharing papers

Overleaf + GitHub Actions + GitHub Pages + Slack

- Overleaf has a good GitHub integration by default
- GitHub Actions can automatically process pushed TeX files
- GitHub Actions can upload the end result to GitHub Pages
- GitHub Actions can post a message to Slack upon completion

https://github.com/arcatdmz/acm2024-primary-article-template

Let's look at the actual code on GitHub:

CI/CD case studies 2:
Running heavy workloads on clouds

Amazon EC2 + GitHub + GitHub Actions

- Automated self-hosted runner management with GitHub personal access token of **`repo`** scope
- Automated EC2 instance management with AWS access keys with permissions for **`ec2:*`** actions
- On-demand tests/experiments with EC2 instance! https://github.com/machulav/ec2-github-runner

CI/CD case studies 3:
ChatGPT integration

GitHub + GitHub Actions + OpenAI Platforms API

For each GitHub comment,

- look for harmful text (moderations endpoint)
  https://github.com/takahiroanno2024/election2024/blob/main/.github/workflows/comment-review.yml#L89-L96
- look for URLs and check if they look harmful
  https://github.com/takahiroanno2024/election2024/blob/main/.github/workflows/comment-review.yml#L54-L87

CI/CD case studies 4:
Publishing web services

To be covered in the next section!

Web technologies for development/public relations

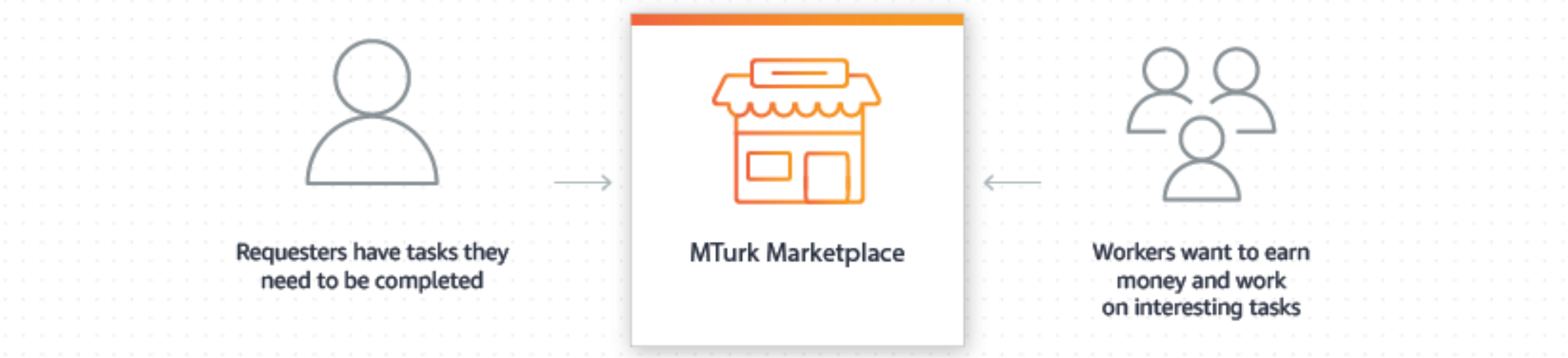
DevOps for research

Web technologies for development/public relations

- Crowdsourcing
- npx + Prettier for code beautification
- Infrastructure for web deployments
- Languages and frameworks
- Tips for making informative portfolio pages

Crowdsourcing

https://www.mturk.com/



- Amazon Mechanical Turk, Yahoo! Crowdsourcing cf. acaric website
- Crowdsourcing and LLMs (search llm modeling human behaviors to replace crowdsourcing or visit HCOMP website)

npx + Prettier for code beautification

- `npx` comes with Node.js by default and Prettier is a well-known OSS code formatter
- `npx prettier --write {your files}` is useful, even if you don't write JavaScript (e.g, HTML and Markdown files; more languages if combined with plugins)
- `npx http-server {your directory}` is also useful
- See also: `localtunnel`, `cpx`, `cross-env`, …

Tell me your favorite `npm` package!

Infrastructure for web deployments

- For static sites: GitHub Pages and Cloudflare Pages
- For a bit more complex sites: Netlify and Vercel
- For backends: ~~PlanetScale~~ TiDB and... Google Sheets!
- For professional-minded developers: AWS, GCP, Azure

Web technologies case studies 1:
Cloudflare Pages + Google Sheets

- Google Forms ➡ Google Sheets
- Google Sheets ➡ Cloudflare Pages

https://github.com/ChezTachibana/website

Let's look at the actual code on GitHub:

Languages and frameworks

- HTML, Pug, Markdown
- React (JSX/TSX), MDX, and more!
- Responsive design for free with CSS frameworks
- Pretty icons with FontAwesome and alternatives

Languages and frameworks

HTML, simplified

Pug

- **html**
  - **body**
    - **h1#mytitle** Main title
    - **p** This is the main paragraph
    - **ol.myclass**
      - **li** This is the first item
      - **li** second item
      - **li** third item
    - **p** You can add
      - **a(href="#mytitle")** links
      - **|** pretty easily!

Markdown

- **#** Main title **{#mytitle}**
- This is the main paragraph
  - **1.** This is the first item
  - **2.** second item
  - **3.** third item
- You can add **[links](#mytitle)** pretty easily!

Languages and frameworks
HTML, evolved

React (*.jsx)

```
// Every user interface component is a function
export function MyButton = () => {
  const handleClick = useCallback(() => { ··· }, []);
  return <Button onClick={handleClick}
className=“my-button">My button!</Button>
}
```

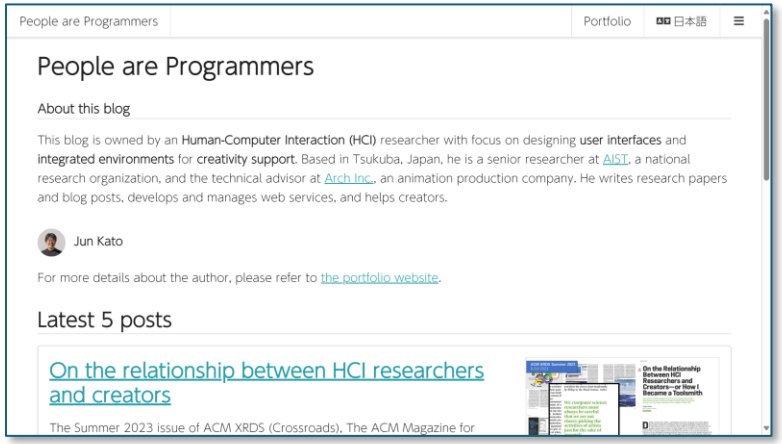
MDX

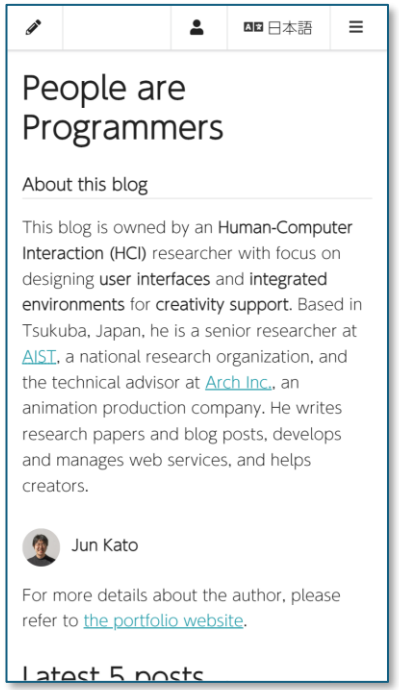
```
import { MyButton } from
"../src/MyButton";

This is the main paragraph
with a custom <MyButton />.
```

Languages and frameworks

CSS frameworks for responsive design

- Bootstrap
- Semantic UI / Fomantic UI
- and more!





Search for “CSS frameworks”

Tips for making informative portfolio pages

- Stick to static sites, allow easy copy and paste
- Show the story, not only the papers
- Single source of truth, multiple views for multiple purposes

Web technologies case studies 2:
Portfolio web page

https://github.com/arcatdmz/arcatdmz.github.io

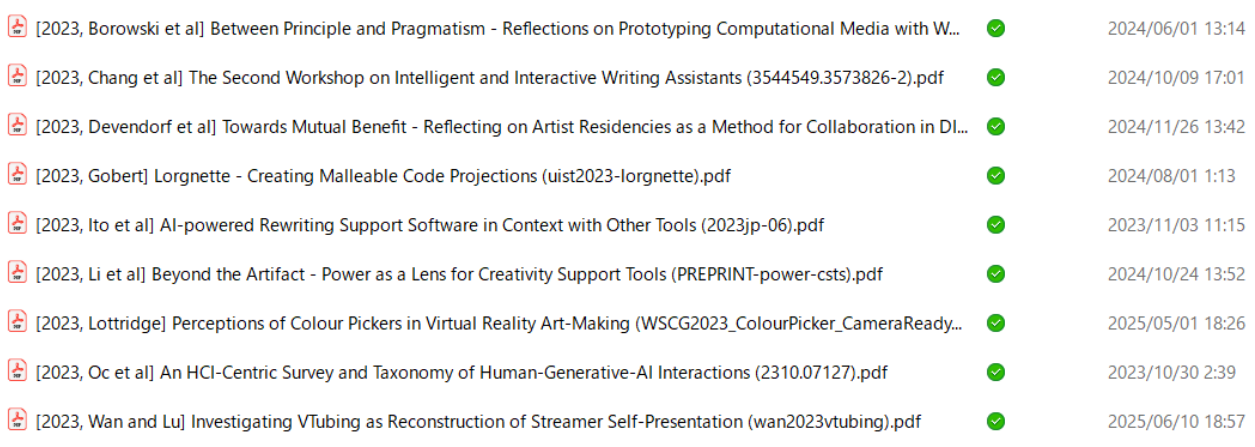
Let's look at the actual code on GitHub:

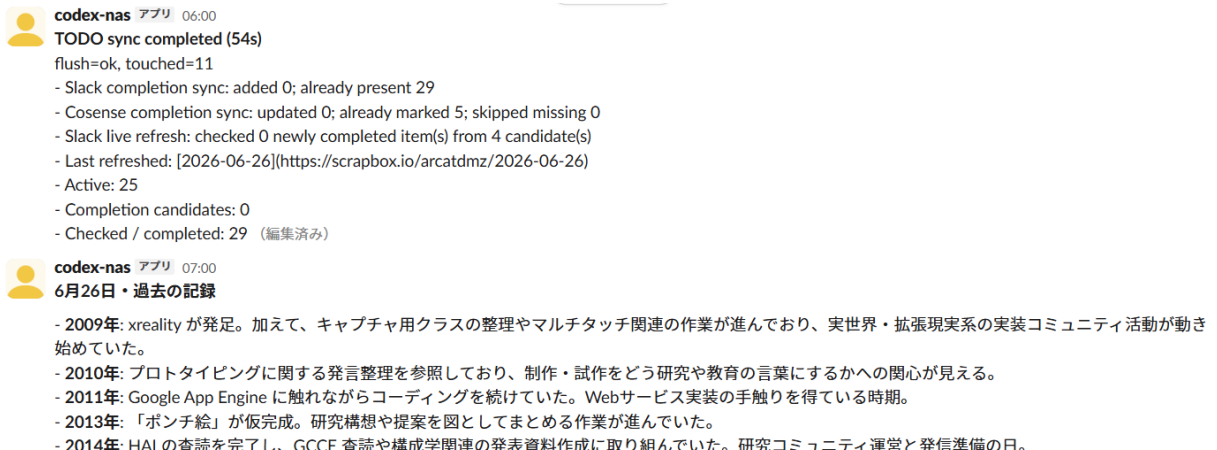
Agentic Development of Your Agentic Assistant

DevOps for Research

Build Your Own Agentic Assistant

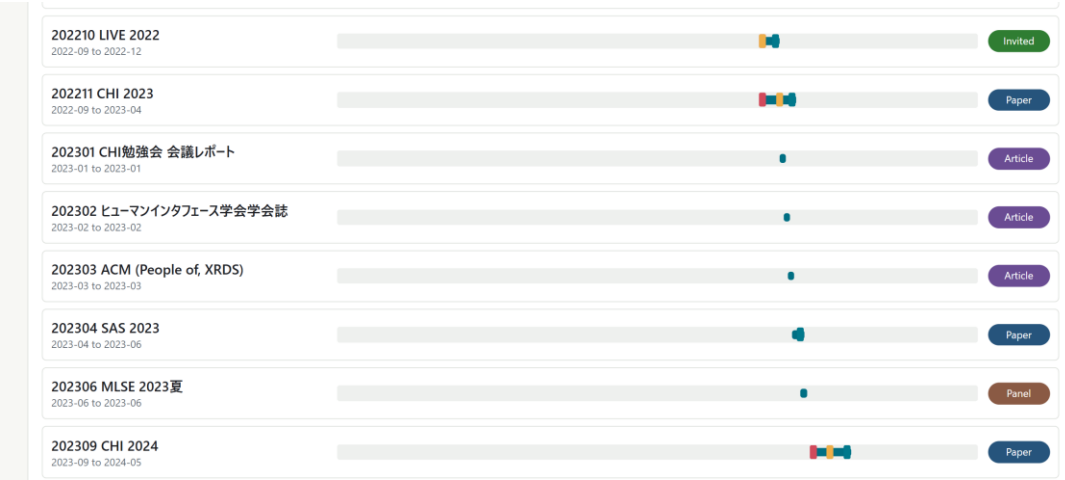
- Organize PDF files
- Filter your X bookmarks to find books to buy
- Plan your day
- Review what you did that day/week/month

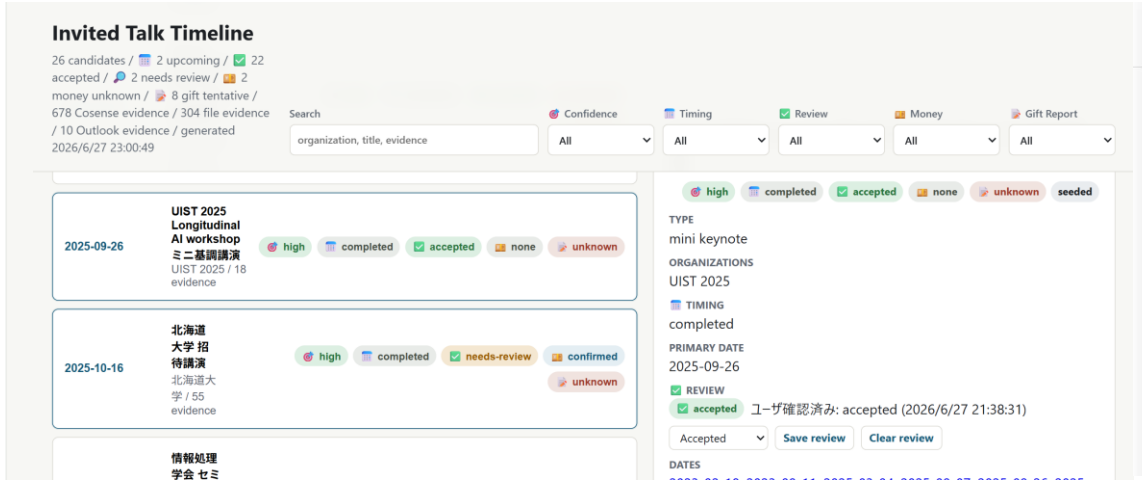


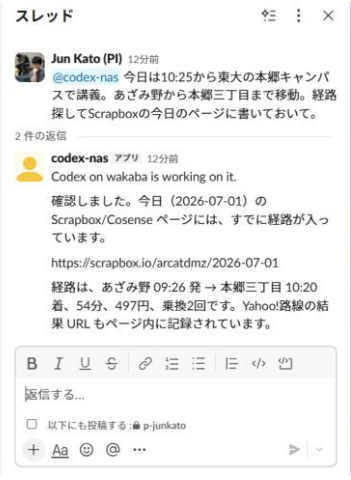


Build Your Own Agentic Assistant

- Add text labels to PDF file(s)
- Manage to-dos across services
- Visualize your paper submission history
- Organize your presentation history
- etc.







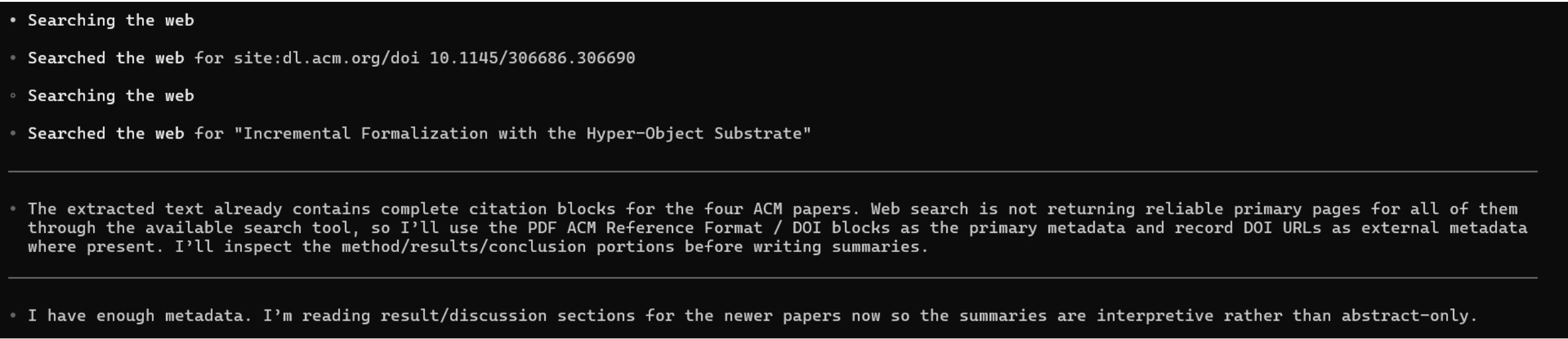
Agentic Development of Your Agentic Assistant

Design Principles

- Rely on interpretive skills when needed
- Single Source of Truth
- Start small and turn repetitive tasks into CLIs
- Record as much as possible; it's like tending bonsai

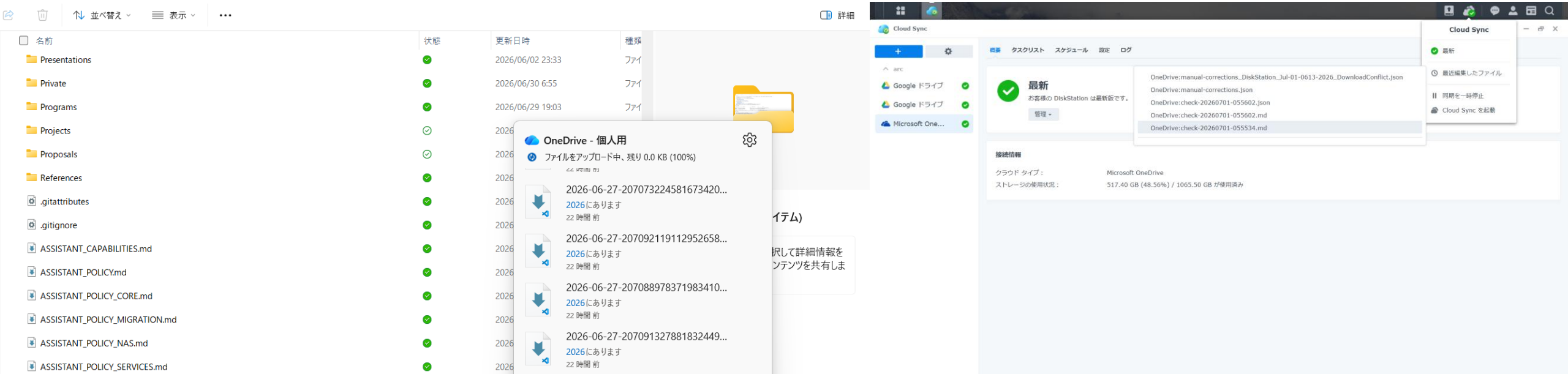
Rely on Interpretive Skills When Needed

- Although Codex is very good at coding, it can also handle interpretive tasks, such as reading papers
- You can even run Codex on your NAS and have it respond to your requests whenever needed



Single Source of Truth

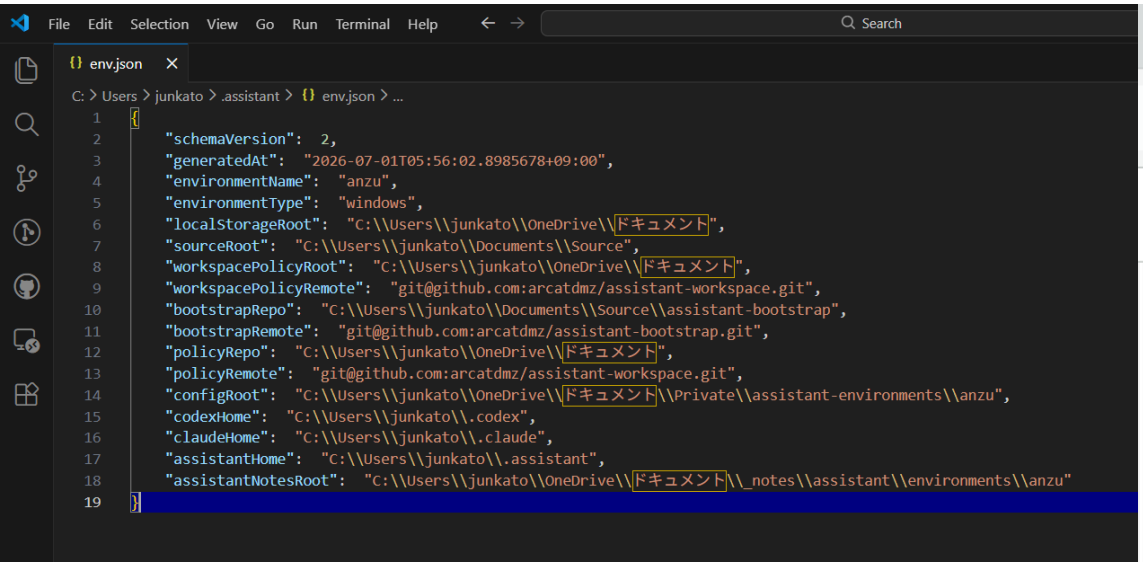
- Store code on GitHub and data in the cloud, and treat them as your single source of truth
- This way, you can run multiple instances of your assistant across different machines and environments

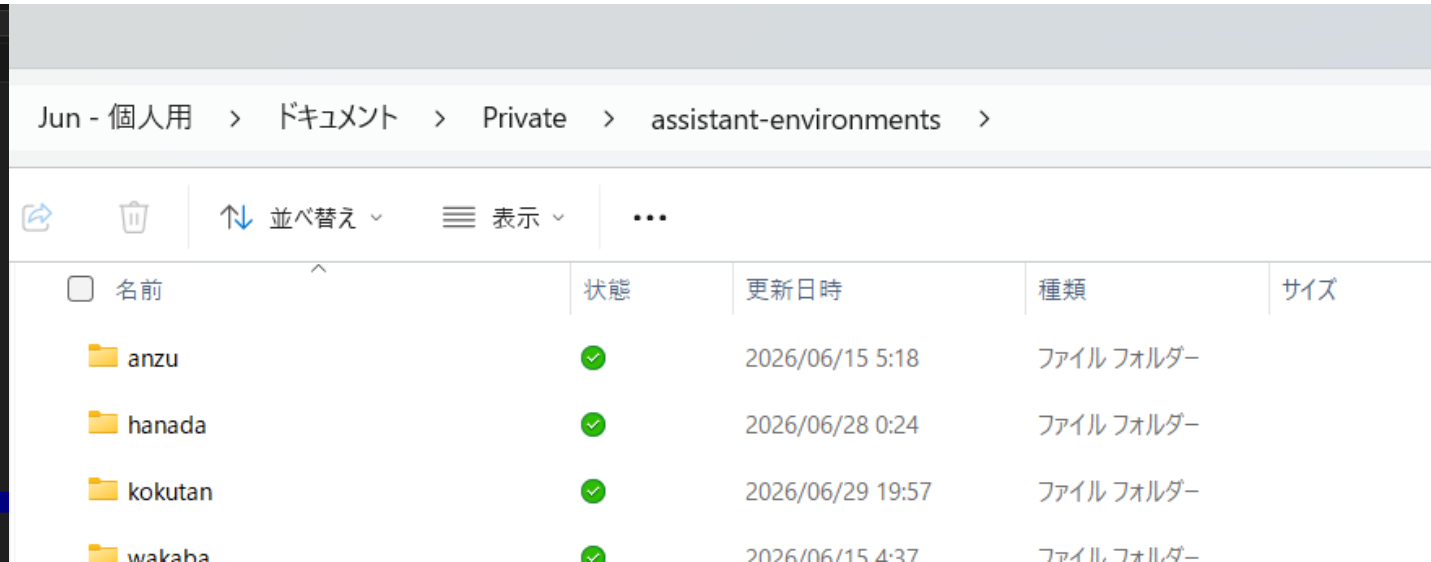


Single Source of Truth / Multiple Instances

- Create a bootstrap document and script

- Use discoverable paths (e.g., ~/.assistant) for config storage
- Use the machine name for machine-specific config paths





Start Small
and Turn Repetitive Tasks into CLIs

- OpenAI Codex: **`codex`**
- Claude Code: **`claude`**

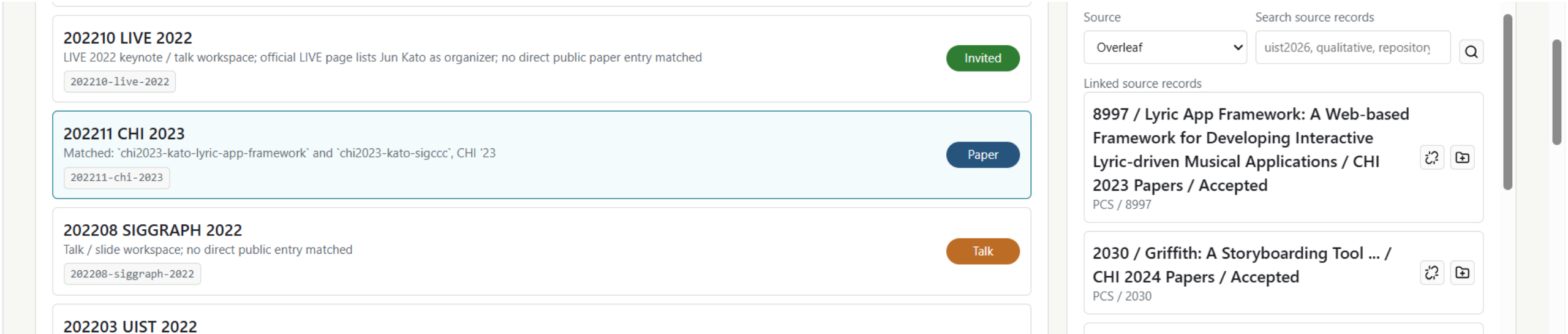
- You can ask anything you would ask in their web interfaces
- "How can I get to Hongo-sanchome Station from …?"

- When you notice that you are asking the same kind of question or performing the same kind of task repeatedly, turn it into a reproducible command-line interface (CLI)

Record as Much as Possible; It's Like Tending Bonsai

- Record your activities for later reuse
- Record agents' notes for transparency and reproducibility

- Ask agents not only to visualize data, but also to implement UIs for verification and modification



Agentic Development of Your Agentic Assistant

Design Principles

- Rely on interpretive skills when needed
- Single Source of Truth
- Start small and turn repetitive tasks into CLIs
- Record as much as possible; it's like tending bonsai

Build Your Own Assistant!

DevOps for research: agenda

- Development environment setup
- Continuous integration/delivery
- Web technologies for development/public relations
- Agentic development of your agentic secretary

DevOps for research