Setup Guide (macOS)

This guide walks you through installing required tools, cloning the repo via SSH, running the site locally to preview edits, and publishing changes to GitHub Pages using GitHub Actions.

> Already have Node.js (via nvm) and Git? Jump to Quickstart — or go to Get the Code and Use the Correct Node Version.

Table of Contents

What You're Building

- The Cognition & Decision Lab website (Astro + React + TailwindCSS)
- Run the site locally, make content updates, and preview changes live
- Publish updates automatically via GitHub Actions to GitHub Pages

Prerequisites at a Glance

- macOS Terminal (default shell: `zsh`)
- Homebrew (package manager)
- Git (via Homebrew or Apple Command Line Tools)
- Node.js managed by nvm (Node Version Manager)
- GitHub SSH access (for cloning/pushing)

Quickstart (after tools are installed)

```
git clone git@github.com:cognition-decision-lab/cognition-decision-
lab.github.io.git
cd cognition-decision-lab.github.io
nvm install && nvm use
npm install
npm run dev
```

Open the printed URL (usually http://localhost:4321). Stop with Ctrl+C.

Next: jump to Common Edits to update people, news, or papers.

---

1) Install Tools

1.1 Homebrew (package manager)

Open Terminal and run:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Follow on-screen steps. If instructed, add the displayed `eval` line to your `~/.zprofile` or `~/.zshrc`, then restart Terminal.

1.2 Install Git

Option A — Homebrew (recommended if you just need Git):

```
brew install git
```

Option B — Apple Command Line Tools (also installs Git and compilers):

```
xcode-select --install
```

Notes:

- Homebrew itself may prompt to install Command Line Tools the first time you use it. Either path is fine.
- For this project, Git is sufficient; compilers are not required.

1.3 nvm (Node Version Manager)

Option A — Install nvm via Homebrew (recommended on macOS):

```
brew install nvm
mkdir -p ~/.nvm
echo 'export NVM_DIR="$HOME/.nvm"' >> ~/.zshrc
echo '[ -s "$(brew --prefix nvm)/nvm.sh" ] && . "$(brew --prefix
nvm)/nvm.sh"' >> ~/.zshrc
source ~/.zshrc
```

Option B — Install nvm via the official script:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.1/install.sh |
bash
echo 'export NVM_DIR="$HOME/.nvm"' >> ~/.zshrc
echo '[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"' >> ~/.zshrc
source ~/.zshrc
```

Verify nvm is installed:

```
command -v nvm
```

You should see `nvm`.

> Tip: If you prefer, you can restart Terminal instead of running `source ~/.zshrc`.

Verify your tools (quick check)

```
git --version
nvm --version
node -v
npm -v
```

If any command errors, revisit the tool's install step above.

---

2) Set up GitHub SSH (required for SSH clone)

If you haven't set up SSH with GitHub yet, do this once:

1. Generate an SSH key (use your GitHub email):

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

If you see an error about ed25519 not supported, use:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

2. Start the SSH agent and add your key (uses macOS keychain):

```
eval "$(ssh-agent -s)"
mkdir -p ~/.ssh
cat << 'EOF' >> ~/.ssh/config
Host github.com
  HostName github.com
  User git
  AddKeysToAgent yes
  UseKeychain yes
  IdentityFile ~/.ssh/id_ed25519
EOF

ssh-add --apple-use-keychain ~/.ssh/id_ed25519 2>/dev/null || ssh-add -K
~/.ssh/id_ed25519
```

3. Add the public key to GitHub:

```
pbcopy < ~/.ssh/id_ed25519.pub
```

- Go to https://github.com/settings/keys → New SSH key → paste → Save.

4. Test your connection:

```
ssh -T git@github.com
```

You should see a success message (e.g., “Hi ! You've successfully authenticated...”). If prompted to continue, type `yes`.

5. Set your Git identity (so commits show the correct author):

```
git config --global user.name "Your Name"
git config --global user.email "your_email@example.com"
```

6. Ensure the remote uses SSH (inside this repo folder after cloning):

```
git remote -v
git remote set-url origin git@github.com:cognition-decision-lab/cognition-
decision-lab.github.io.git
```

7. (Optional) You can skip test pushes; you only need SSH for cloning/pulling/pushing your actual changes later.

---

3) Get the Code

Choose a folder **NOT synced by iCloud/Dropbox/Google Drive** to avoid file-watcher slowness. For example `~/code`:

```
cd ~
mkdir -p code && cd code
git clone git@github.com:cognition-decision-lab/cognition-decision-lab.github.io.git
cd cognition-decision-lab.github.io
```

---

4) Use the Correct Node Version

This project pins Node version in `.nvmrc` to `v22.11.0`. Let nvm install and activate it:

```
nvm install
nvm use
node -v
npm -v
```

- `node -v` should print `v22.11.0`.
- Optional: make Node 22 your default:

```
nvm alias default 22
```

---

5) Install Dependencies

From the project folder:

```
npm install
```

---

6) Run the Website Locally

Start the dev server:

```
npm run dev
```

Open the printed URL in your browser (Astro usually runs at `http://localhost:4321`).

Stop the server any time with `Ctrl+C` in that Terminal window.

> If the port is busy, you can change it:
>
> ```
> npm run dev -- --port 4322
> ```

(Optional) Edit in VS Code

- Download VS Code: https://code.visualstudio.com
- Open this project folder
- Recommended extensions:
  - Astro
  - Tailwind CSS IntelliSense
  - ESLint

Troubleshooting

- “command not found: nvm”
  - Run `source ~/.zshrc` and try again.
  - If still failing, re-run the nvm install script, then restart Terminal.
- Node version doesn’t match
  - Inside the project folder, run:

    ```
    nvm install && nvm use
    ```
- npm permission errors (EACCES)
  - Avoid `sudo` with npm. Ensure you’re using nvm and the correct Node:

    ```
    nvm use
    ```
- Dev server won’t start, port in use

```
npm run dev -- --port 4322
```

- Firewall prompt on first run
  - Click “Allow” for incoming connections.
- Slow or constant rebuilds
  - Move the project out of cloud-synced folders to a regular local folder like `~/code`.
- Install failures (network)
  - Ensure internet is working.
  - Set npm registry and retry:

```
npm config set registry https://registry.npmjs.org/
npm install
```

Project Structure

- Astro app with scripts in `package.json`:
  - `dev` — start local dev server
  - `build` — production build to `dist/`
  - `preview` — preview the production build
- Source code in `src/` and static assets in `public/`

Common Edits (Content)

- People: edit `src/data/people.json`
  - Sections: `faculty`, `researchers`, `students`, and `alumni`.
  - Images: place files under `public/faculty/`, `public/researchers/`, or `public/students/` and reference with a leading slash, e.g. `/students/jane_doe.jpg`.
  - Fields typically include `name`, `image`, `title`, and optional `website`, `linkedin`, `email`.
- News: edit `src/data/news.json`
  - Each item has `date` and `text`.
  - `text` supports simple HTML links and emphasis if needed.
- Papers: edit `src/data/papers.json`
  - Common fields: `title`, `authors` (array), `journal`, `date`, `type` (`published` or `working`), `section`, `url`.
  - Optional fields: `volume`, `issue`, `pages` where applicable.

- Images and assets
  - Add images to `public/` in the appropriate subfolder (e.g., `public/students/your_file.jpg`).
  - Use web-friendly formats (`.jpg`, `.jpeg`, `.png`, `.webp`) and keep file sizes modest for fast loads.
- Verify locally
  - Run:

```
npm run dev
```

  - Open the local URL, click through People, News, and Papers to confirm your changes.

Contributing (Git Basics)

- Create a branch for your change:

```
git checkout -b feature/short-description
```

- Stage and commit:

```
git add -A
git commit -m "Update people: add Jane Doe"
```

- Push your branch and open a Pull Request (recommended), or push to `master` if that's your workflow:

```
git push -u origin feature/short-description
```

- After merge or push to `master`, GitHub Actions will build and deploy automatically (see Publish section).

Notes:

- Do not commit `node_modules/` or build outputs (`dist/`) — they're ignored by `.gitignore`.
- Keep commit messages clear and concise.

Publish the Website

This repository already has GitHub Pages deployment configured via Actions.

GitHub Actions (Already Set Up — Recommended)

- Workflow file: `.github/workflows/deploy.yml` (runs on pushes to `master` and via manual trigger)
- Live site: https://cognition-decision-lab.github.io/

Trigger a deployment by either:

1. Pushing to `master`:

```
git add -A
git commit -m "Update site content"
git push origin master
```

2. Manually from GitHub UI:

- Go to the repository → Actions → “Deploy to GitHub Pages”
- Click “Run workflow” → select branch `master` → Run

Check status:

- Actions runs: https://github.com/cognition-decision-lab/cognition-decision-lab.github.io/actions
- Deployed site: https://cognition-decision-lab.github.io/

Notes:

- The workflow uses `withastro/action@v4`, which installs dependencies, builds, and publishes automatically.
- `src/settings.ts` sets `template.website_url` to the production URL and `template.base` to `` (empty), which is correct for a user/org `*.github.io` site.
- If you later deploy a project site (not `*.github.io`), set `template.base` to `"/REPO_NAME"`.

Next Steps

- To exit the dev server: press `Ctrl+C` in the Terminal
- To build for production:

```
npm run build
npm run preview
```

- To update dependencies later:

```
npm install
```

Acknowledgements

This website is built on the excellent Astro Academia template by Maio Barbero.

- Base template: https://github.com/maiobarbero/astro_academia

We appreciate the open-source community behind Astro, Tailwind CSS, and DaisyUI.

Windows/Linux Appendix (Basics)

Windows — Option A: nvm-windows (No WSL)

- Install Git for Windows: https://git-scm.com/download/win
- Install nvm-windows: https://github.com/coreybutler/nvm-windows/releases
- In a new Command Prompt or PowerShell:

```
nvm install 22.11.0
nvm use 22.11.0
node -v
npm -v
```

- SSH key (use Git Bash):

```
ssh-keygen -t ed25519 -C "your_email@example.com"
cat ~/.ssh/id_ed25519.pub
```

  Add the key in GitHub → Settings → SSH and GPG keys. Test:

```
ssh -T git@github.com
```

- Clone and run (Git Bash or PowerShell):

```
git clone git@github.com:cognition-decision-lab/cognition-decision-
lab.github.io.git
cd cognition-decision-lab.github.io
npm install
npm run dev
```

Windows — Option B: WSL Ubuntu (Recommended)

- Install WSL + Ubuntu (PowerShell as Admin):

```
wsl --install -d Ubuntu
```

Restart if prompted, then open “Ubuntu” from Start.

- Install Git and curl:

```
sudo apt update && sudo apt install -y git curl ca-certificates
```

- Install nvm + Node:

```
curl -o- https://raw.githubusercontent.com/nvm-
sh/nvm/v0.40.1/install.sh | bash
echo 'export NVM_DIR="$HOME/.nvm"' >> ~/.bashrc
echo '[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"' >> ~/.bashrc
source ~/.bashrc
nvm install 22.11.0
nvm use 22.11.0
```

- SSH key (inside Ubuntu):

```
ssh-keygen -t ed25519 -C "your_email@example.com"
eval "$(ssh-agent -s)"
mkdir -p ~/.ssh && chmod 700 ~/.ssh
printf "Host github.com\n  HostName github.com\n  User git\n
AddKeysToAgent yes\n  IdentityFile ~/.ssh/id_ed25519\n" >>
~/.ssh/config
ssh-add ~/.ssh/id_ed25519
cat ~/.ssh/id_ed25519.pub
```

Copy the key output → GitHub → Settings → SSH and GPG keys → New SSH key.

- Clone and run (inside Ubuntu):

```
git clone git@github.com:cognition-decision-lab/cognition-decision-
lab.github.io.git
cd cognition-decision-lab.github.io
nvm install && nvm use
npm install
npm run dev
```

WSL — Space & safety notes (brief)

- Typical initial disk use: a fresh Ubuntu distro is ~0.5–2 GB. WSL2 stores the filesystem in a dynamically-growing virtual disk (ext4.vhdx) under your Windows profile.
- Plan for additional space: allow at least **8–10 GB** free for development (Node packages, caches, and builds can grow quickly).

- Safety: WSL is sandboxed and does not modify Windows system files or the bootloader. Installing WSL enables optional Windows features (WSL and Virtual Machine Platform) and may require a reboot.
- Resource control: you can limit WSL VM memory/CPUs by creating `C:\Users\\<you>\\.wslconfig` with a `[wsl2]` section (example below).

Example `.wslconfig` to limit resources:

```
[wsl2]
memory=4GB
processors=2
swap=1GB
```

- Checking space: inside WSL run `df -h` and `du -sh ~` or on Windows inspect the distro VHDX under `%LOCALAPPDATA%\Packages\`.
- Reclaim or move space: export/import the distro (`wsl --export` / `wsl --import`) or `wsl --unregister` to remove it. Compacting the VHDX is also possible via Hyper-V tools.
- Why WSL is more reliable for development: it provides a real Linux environment (bash, apt, nvm, Linux kernel), avoids Windows-specific path/permission quirks, preserves file-watcher performance when projects are stored inside the WSL filesystem, and integrates cleanly with VS Code Remote - WSL for an editor experience that runs inside Linux.

Linux (Ubuntu/Debian)

- Install Git and curl:

```
sudo apt update && sudo apt install -y git curl ca-certificates
```

- Install nvm + Node:

```
curl -o- https://raw.githubusercontent.com/nvm-
sh/nvm/v0.40.1/install.sh | bash
echo 'export NVM_DIR="$HOME/.nvm"' >> ~/.bashrc
echo '[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"' >> ~/.bashrc
source ~/.bashrc
nvm install 22.11.0
nvm use 22.11.0
```

- SSH key:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
eval "$(ssh-agent -s)"
mkdir -p ~/.ssh && chmod 700 ~/.ssh
```

```
printf "Host github.com\n  HostName github.com\n  User git\n
AddKeysToAgent yes\n  IdentityFile ~/.ssh/id_ed25519\n" >>
~/.ssh/config
ssh-add ~/.ssh/id_ed25519
cat ~/.ssh/id_ed25519.pub
```

- Clone and run:

```
git clone git@github.com:cognition-decision-lab/cognition-decision-
lab.github.io.git
cd cognition-decision-lab.github.io
nvm install && nvm use
npm install
npm run dev
```

If you get stuck, copy the exact error message from Terminal and ask for help.