Tool setup: shell, gh, and GitHub identity

Key takeaways

1. Run gh where course work runs; approve the login in a local browser.
2. Verify the exact GitHub login that will own your private course repositories.
3. Use gh to inspect repositories, pull requests, and checks as terminal evidence.

Common misunderstandings of GitHub CLI setup

Which claim would you defend right now? Choose before the evidence.

"Browser approval signs in my laptop, not the remote shell running gh."

"My UT EID or GitHub display name is enough to seed the correct repository."

"gh is only for cloning; pull-request and CI state still require a browser."

Keep your choice. Look for the mechanism or counterexample that would make it fail.

Your agent inherits its environment

The shell supplies its tools, paths, quota, and credentials

One missing tool, and the agent starts changing the machine

```
> Run the tests before we change anything.

$ pytest -x
zsh: command not found: pytest

pytest is missing here — I'll install it system-wide
with sudo pip so it stays available...
```

- Every tool call runs in your shell: your PATH, your quota, and your credentials.
- Today’s shared goal is a known-good shell and verified account identities — not a completed homework.
- Your private repository becomes the working environment after Canvas receives your GitHub login.

Department Linux: the setup we know works

Rather set up your own laptop directly? Go right ahead

many

department Linux hosts

1

shared file system across all department hosts

15 GB

undergrad quota: check early

- SSH from any laptop into a department host with git, gh, tmux, node, and rg already present.
- All department hosts share one home directory, so choose any healthy host from the status page.
- Run /lusr/bin/chkquota before later installing the course toolchain; the dotfiles setup requires 2 GB free.
- On your own machine the steps are the same; you install git and gh and fix anything that differs.

The command is remote; the approval is local

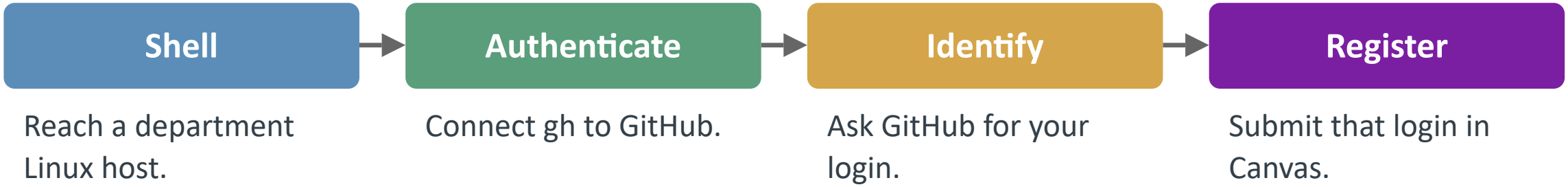
Run gh on the department host and approve its browser flow on your laptop

Where	What happens	Why it matters
Department host	gh auth login starts a one-time browser flow	The CLI is authenticated where your course work will run
Laptop browser	Sign into the intended GitHub account and approve the code	The department host does not need a graphical browser
Department host	Return to the shell and verify the account	Only then copy the login that Canvas will receive

- Keep the SSH session open while you complete the browser approval.
- The GitHub account approved in the browser is the account gh will use on the host.
- Your UTCS account reaches the host; your GitHub account owns the course repositories.

In class: register your GitHub identity

One verified identity connects your shell, GitHub, Canvas, and private repositories



- The GitHub CLI and Canvas submission must show the same login.
- Repository creation happens after the submission is processed; an invitation may arrive after class.
- Do not create a substitute repository or fork the public dotfiles repository while you wait.

Exercise part 1: reach the known-good shell

This establishes a common starting point for the room

```
# Pick a healthy machine from the CS host-status page
ssh <csid>@<host>.cs.utexas.edu
hostname -f          # confirm which host you landed on
/lusr/bin/chkquota   # confirm at least 2 GB free

mkdir -p ~/utac && cd ~/utac
```

- Your UTCS account is only for the department host; it is not necessarily your GitHub login.
- ~/utac will hold ordinary course repositories. The later dotfiles checkout belongs at ~/dotfiles.
- Continue only with at least 2 GB free; otherwise resolve SSH or quota before debugging GitHub.

gh

GitHub as commands: authentication, repositories, PRs, checks, and API JSON.

Replaces: browser-only GitHub inspection

```
# Which account is this shell using?
gh auth status

# Copy a repository onto this host
gh repo clone OWNER/REPO

# Follow a PR's CI checks until they finish
gh pr checks --watch
```

Exercise 2: authenticate and register

Run gh in the SSH session; use your laptop browser only for approval

The host displays a one-time code; the laptop browser approves it

```
gh auth login
# Choose: GitHub.com -> HTTPS -> Login with a web browser.
# Copy the one-time code; press Enter on this host.
# On your laptop, open github.com/login/device and enter the code.

gh auth status                  # confirm the intended account
gh api user --jq .login        # prints the Canvas value
```

- Submit only the final command's output — no @, URL, EID, or email address.
- Confirm the Canvas value, then stop; the repository may arrive later.

GitHub and agent accounts are separate

Each service answers a different question

Service	Account to use	What to verify today
GitHub / gh	Your GitHub account	gh api prints the login submitted to Canvas
Codex / ChatGPT	UT ChatGPT account	You can sign in at chatgpt.com
Claude Code	UT Austin Claude workspace	You can sign in at claude.ai

- Browser access is enough today; CLI login happens after your private dotfiles setup installs the tools.
- The schedule links full account-setup reference decks for Codex and Claude Code.

Public is the template; private is your checkout

Public course updates flow into a private repository that holds your work

Repository	Purpose	What you do
utac-f26/dotfiles	Public template and update source	Browse it; use it for reference
utac-f26/dotfiles-<login>	Your private homework repository	Clone, edit, commit, and push

Do not fork the public repository or clone it to ~/dotfiles. The setup script refuses a public origin so homework cannot be pushed there by mistake.

After your private repository arrives

The invitation may arrive after class; nothing is wrong while the seeder catches up

Run these commands only after the private repository exists

```
gh repo clone utac-f26/dotfiles-<login> ~/dotfiles
cd ~/dotfiles
./setup
```

- Accept the GitHub invitation if GitHub asks; then clone the private repository to exactly ~/dotfiles.
- Setup configures ordinary git pull to receive public course updates and git push to publish only to your private repository.
- The assignment specification owns the remaining login, verification, policy, and submission work outside class.

Course updates arrive by git pull

Setup points pull at the public template and push at your private repository

Setup wired two remotes with different jobs

```
$ git remote -v
origin    utac-f26/dotfiles-<login>   your private repository
upstream  utac-f26/dotfiles           the public course template

$ git pull    # reads upstream: course fixes reach you
$ git push    # writes origin: your work stays private
```

- Setup wrote this configuration; do not re-point the remotes by hand.
- Run git pull when an update is announced; the announcement says whether ./links or ./setup --update is also needed.
- cat KIT_REV prints the kit revision you actually hold, so you can check it against the one announced.

Dotfiles merges; homework pulls may rebase

The update source determines the pull policy

State	Pull behavior	Result
No commits after the remote tip	Fast-forward	Update lands without a merge
Dotfiles: public update plus pushed work	Merge	Private history stays unchanged
Homework: course beat plus post-pin work	git pull --rebase	Only unpushed commits replay

Never rebase or force-push commits already on main. Rebase only unpushed post-pin work during a homework pull.

gh turns remote GitHub state into terminal evidence

Use focused fields and exit status instead of reading a browser page by eye

Identity and repository

```
gh auth status
gh api user --jq .login

gh repo view --json
nameWithOwner,visibility
```

Pull request and checks

```
gh pr create --fill --base main

gh pr view --json state,reviewDecision
gh pr checks --watch
```

When a subcommand stops short, ask the API for the one field you need

```
gh api repos/{owner}/{repo}/pulls/42/files --jq '.[].filename'
```

Four first-hour failures

Check the identity or repository boundary before changing tools

Symptom	Likely cause	First check
gh shows the wrong account	Stale GitHub authentication	gh auth status; log in again
Canvas has a URL or @-name	Wrong registration value	Resubmit gh api user --jq .login
Private repo is absent	Seeder or invitation is pending	Wait for the GitHub invitation
Setup rejects public origin	Public repo was cloned	Clone dotfiles-<login> instead

The public dotfiles repository remains available in a browser while private repositories are seeded.

Key Takeaways

1. Run gh on a department host and complete its device approval in your laptop browser.
2. Submit the exact login printed by gh to Canvas, then wait for your private repository.
3. Use gh to clone and inspect repository, pull-request, and check state from the terminal.