Troubleshooting

This page collects common setup problems in IC101 and the shared workspace, including the ones that come up when you load a later course with the workspace's `mod` command. Each entry gives the symptom you'll see, what causes it, and how to fix it.

I don't know how to open a terminal

Symptom: A page says "run this command" and you're not sure where to type it.

Cause: Every lab assumes you already have a terminal window open to type commands into.

Fix: Lab 01, Part 1 walks you through opening one, with steps for each OS.

git: command not found

Symptom: Running `git clone ...` prints `git: command not found` (on Windows, `'git' is not recognized...`) instead of downloading anything.

Cause: git isn't installed yet, or, on Windows, the terminal was already open when you installed it.

Fix: Follow Lab 01, Part 2, then open a new terminal window and try again. On Windows you need the new window, because a terminal that was already open won't pick up a program you installed after it started.

Docker won't start or hello-world fails

Symptom: Docker Desktop never shows **Engine running**, or `docker run --rm hello-world` prints an error instead of `Hello from Docker!`.

Cause: Docker hasn't finished starting, or a one-time OS setting hasn't been applied: WSL2 on Windows, or `docker` group membership on Linux.

Fix:

- Wait until Docker Desktop shows **Engine running** before you run any `docker` command. It can take a minute to get there after you open it.
- On Windows, make sure WSL2 is installed and selected as Docker's backend. See What Docker needs.
- On Linux, check that your user is in the `docker` group: run `groups` and look for `docker` in the output. If you've only just added yourself to the group, log out and back in so that it takes effect.
- Free up disk space. When the disk is full, large image pulls fail without a clear error, so a full disk is easy to miss.

permission denied on a docker command (Linux)

Symptom: A `docker` command fails with a message containing `permission denied`.

Cause: Your Linux user isn't in the `docker` group yet, so it can't talk to Docker without `sudo`.

Fix:

```
sudo usermod -aG docker $USER
```

Then log out and log back in (group changes don't apply to a terminal that's already open), and run the command again.

Port 80 already in use

Symptom: `start_vnc` fails, or the browser won't load `http://localhost/`, because another program on your computer is already using port 80.

Cause: Only one program can listen on a port at a time, and something else, often another web server running on your computer, started listening on port 80 first.

Fix: Start the workbench on a different host port instead:

```
HOST_PORT=8080 ./scripts/start_vnc.sh
```

Then open `http://localhost:8080/`.

Container starts but the browser page is blank

Symptom: `start_vnc` finishes without errors, but `http://localhost/` shows a blank page or won't connect.

Cause: The script exits before the desktop inside the container has finished starting, so for a short while there's nothing for the browser to show.

Fix:

- Wait about 30–60 s after the first pull or start for the services to come up, then reload the page.
- Run `docker ps` and check that it lists a container named `asic-edu-osic`.
- Read that container's logs with `docker logs asic-edu-osic`.
- Check that the address uses the port you actually set: `80` by default, or your `HOST_PORT` if you set one.

.designinit: No such file or directory

Symptom: Loading the environment with `. /foss/designs/common/.designinit` prints `bash: /foss/designs/common/.designinit: No such file or directory`.

Cause: Your `workspace` folder isn't mounted at `/foss/designs`, so the container has only an empty folder there. That happens when the container wasn't started with `start_vnc` from your `workspace` clone: for example, when someone started it with a plain `docker run` of the `hpretl/iic-osic-tools` image.

Fix:

1. In the desktop terminal, run `ls /foss/designs/`. If the workspace is mounted, the list includes `common`, `modules` and `scripts`, among others. If it lists nothing, the workspace isn't mounted.
2. If you haven't cloned the `workspace` repo yet, do Lab 03 from the start. If you have, start the workbench again from your clone, as in Lab 03, Part 2: `./scripts/start_vnc.sh` on macOS or Linux, `scripts\start_vnc.bat` on Windows.

mod: command not found

Symptom: A `mod` command prints `bash: mod: command not found`.

Cause: `mod` is defined in `/foss/designs/common/.designinit`, and this terminal hasn't loaded that file yet.

Fix: Load it, then run your `mod` command again:

```
. /foss/designs/common/.designinit
```

If that prints `No such file or directory`, see .designinit: No such file or directory.

Smoke test: tool or helper not found

Symptom: `smoke_test.sh` prints `FAIL` for a command (for example `ngspice` or `magic`) or for one of the module helper lines (`mod`, `add_module.sh`, `ic101_setup`) instead of `OK`.

Cause: The smoke test loads `common/.designinit` itself before it checks anything, so a terminal that hasn't loaded the file can't make a check fail. The most it can cause is the `WARN  your shell started on PDK ...` line described in Smoke test: PDK is not sky130A. A `FAIL` on a module helper line means the script couldn't find your workspace at `/foss/designs`, where those checks look for `scripts/add_module.sh` and `modules/ic101_setup`. A `FAIL` on a

tool means the tool isn't on the `PATH`, which the image sets up in every terminal, so the container is probably not running the image the scripts expect.

Fix: Loading the environment by hand is still a quick first check, because it fails with `No such file or directory` when the workspace isn't mounted:

```
. /foss/designs/common/.designinit
```

Then check that the workspace is mounted (`ls /foss/designs/scripts` should list files) and run `/foss/designs/scripts/smoke_test.sh` again. If that folder is empty or missing, see .designinit: No such file or directory. If tools are still missing after that, your image may not be the one the scripts expect, so re-pull the tag they use (`DOCKER_TAG`, which defaults to `2026.08`).

Smoke test: PDK is not sky130A

Symptom: The smoke test's PDK line starts with `FAIL  PDK is` or `FAIL  PDKPATH is`, or else the checks pass but one line reads `WARN  your shell started on PDK ....`

Cause: The PDK is there, since SKY130 ships inside the image at `/foss/pdks/sky130A`, but the image starts on a different PDK, `ihp-sg13g2`, so `PDKPATH` and the ngspice and KLayout search paths all point at that one instead. The `WARN` means the smoke test corrected the environment for its own run, but any tools you start from that terminal will still use the wrong PDK.

Fix: Load the workspace environment, which switches all of those variables over to SKY130, and then run the smoke test again:

```
. /foss/designs/.designinit
/foss/designs/scripts/smoke_test.sh
```

All `/foss/designs/.designinit` does is load `/foss/designs/common/.designinit`, so you can use either path.

If you want to switch PDK by hand, the image has its own switcher, `sak-pdk-script.sh`, and you must **source** it, with a leading `.`:

```
. sak-pdk-script.sh sky130A
```

If you run it as a plain command instead, it prints the settings but changes nothing in your shell, which looks a lot like it worked. Its usage message calls it `sak-pdk`, but there's no command by that name on the `PATH`, so type the full `sak-pdk-script.sh`.

mod: no such module

Symptom: `mod <course>` prints `mod: no such module: <course>`, followed by `Available modules:` and a list of the modules you do have.

Cause: There's no folder for that course under `/foss/designs/modules/` yet. Usually that's because you ran `mod <course>` before `mod add <course>`, or because the name is misspelled.

Fix: Check the name against the list it printed. If the course isn't in the list, add it with `mod add <course>`, using the name your course's Lab 01 gives. `mod add` clones the course and moves you into its folder, and from then on `mod <course>` works.

mod add asks for a GitHub username

Symptom: `mod add <course>` stops at `Username for 'https://github.com':` and waits, or it fails with `fatal: could not read Username for 'https://github.com'`, followed by `add_module: clone failed for <course>`.

Cause: GitHub wouldn't let git read `github.com/uoftasic/<course>` without logging in. `mod add` is meant to work without a login, so either the course name is misspelled or that course's repo isn't published yet.

Fix:

- Don't type a username or password. If git is waiting at the prompt, press **Ctrl+C** to stop it. Stopping it there leaves nothing behind, so you can run `mod add` again straight away.
- Check the spelling of the course name and run `mod add <course>` again.
- If the name is right, open `https://github.com/uoftasic/<course>` in your browser. A 404 page there means the repo isn't published yet. Whether or not you see a 404, ask on the team Discord, because only the course team can fix it.

Copy and paste into noVNC

Symptom: `Ctrl+V` on your laptop's keyboard doesn't paste into the noVNC desktop.

Cause: noVNC streams a remote desktop into your browser, and your laptop's clipboard isn't shared with that desktop automatically.

Fix: Press **Ctrl+Shift+V** instead of `Ctrl+V` to paste into the desktop, or use the clipboard icon in the noVNC sidebar.

Still stuck

If none of these fit, ask on the team Discord. Include your OS, your Docker version, the exact command you ran, and the part of the log that shows the error, but leave out passwords and other secrets.