

# How to prepare a quality screen-share

## 📌 Objectives

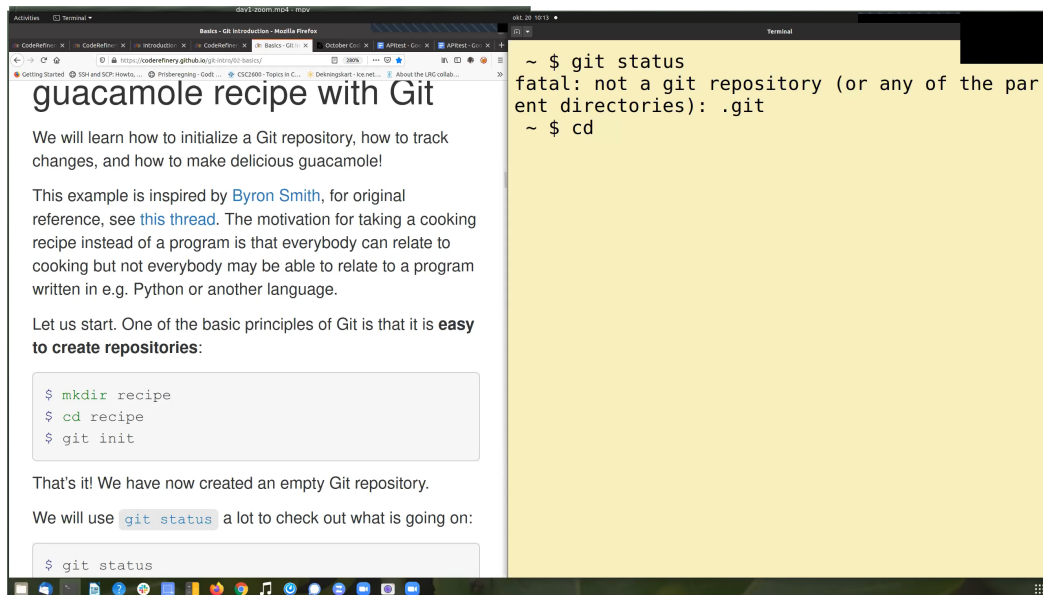
- Discuss the importance of a well planned screen-share.
- Learn how to prepare and how to test your screen-share setup.
- Know about typical pitfalls and habits to avoid.

## Instructor note

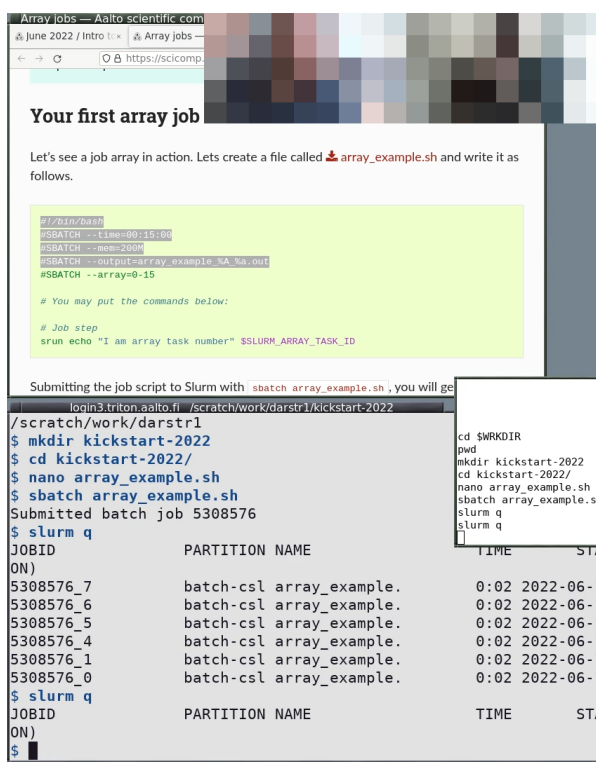
- Discussion: 15 min
- Exercises: 15 min

## Share portrait layout instead of sharing entire screen when teaching online

- Many learners will have a smaller screen than you.
- You should plan for learners with **only one small screen**.
- A learner will **need to focus on both your screen share and their work**.
- Share a **portrait/vertical half of your screen** (840 × 1080 is our standard and your maximum).
- Zoom provides a “Share a part of screen” that is good for this.
  - Our latest streaming setup (day 4) can take the portrait part for you so you can share landscape.
  - Zoom + Linux + Wayland display manager doesn’t have “Share a portion of the screen”
    - You can share a single window portrait.
    - Or you can start your desktop session in “X11” or “Xorg” legacy mode.
    - Or possibly other workarounds (does anyone know other solutions?)



A FullHD 1920x1080 screen shared. Learners have to make this really small if they want to have something else open.

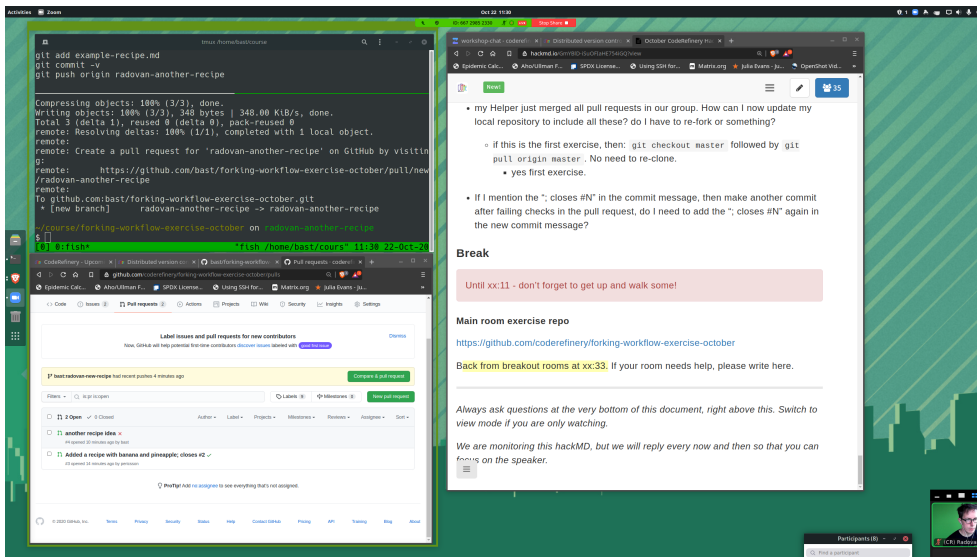


Portrait layout. Allows learners to have something else open in the other half.

Motivation for portrait layout:

- This makes it easier for you to look at some other notes or to coordinate with other instructors at the same time without distracting with too much information.
- This makes it possible for participants to have something else open in the other screen half: terminal or browser or notebook.

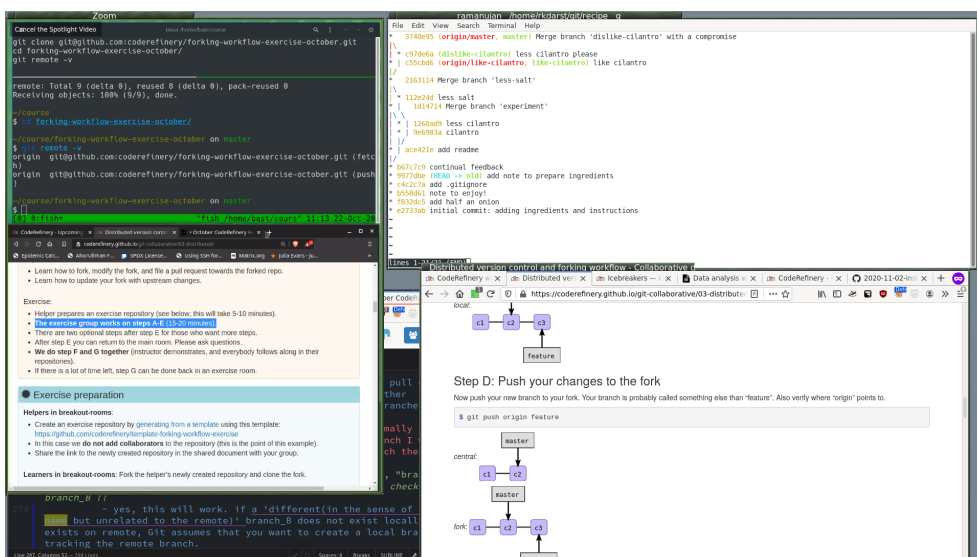
## Instructor perspective



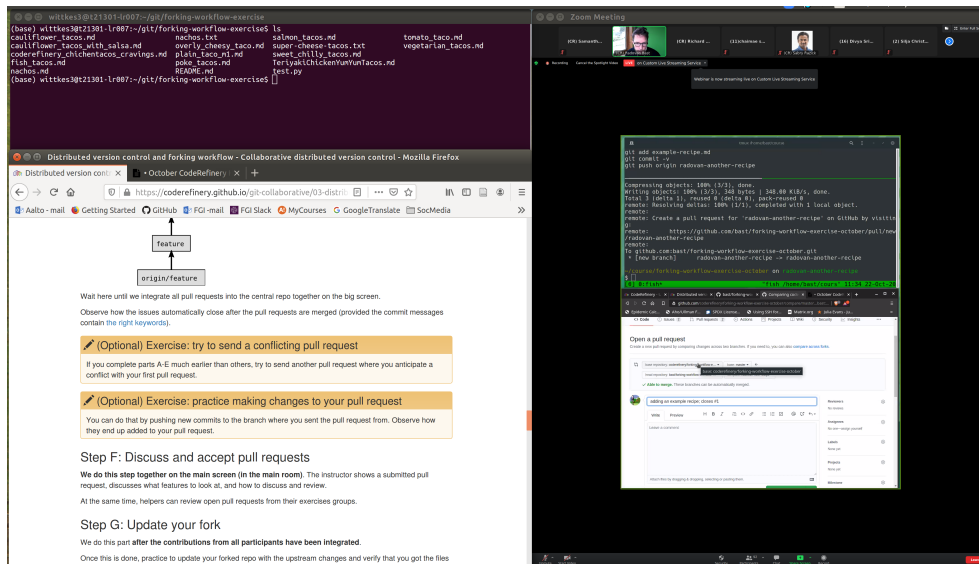
**I1:** This is how it can look for the instructor. Zoom is sharing a portion of the left side, the right side is free for following notes, chat, etc.

## Learner perspective

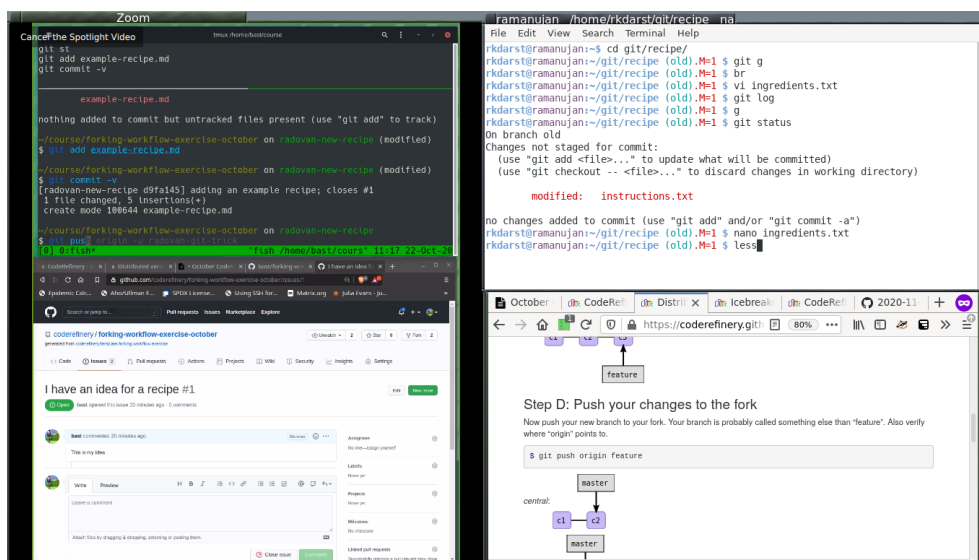
Here are three examples of how it can look for the learner.



**L1:** Learner with a large screen, Zoom in dual-monitor mode so that the instructor pictures are not shown. Screen-share is on the left side, collaborative notes at bottom left, terminal and web browser on the right.



**L2: A learner with a single large screen (Zoom in "single monitor mode"). Instructor screen share at right, learner stuff at left.**



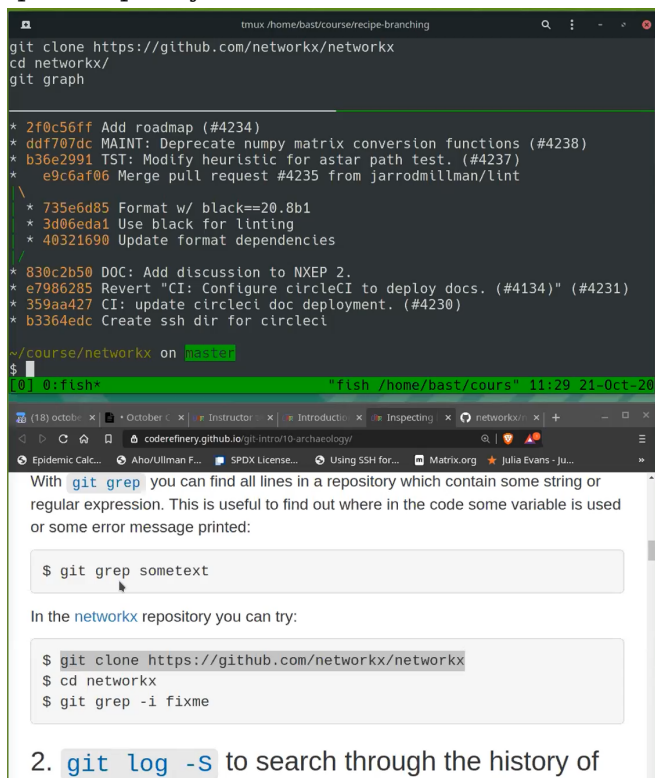
**L3: A learner with a particularly small screen. Instructor screen-share at left, your windows at right.**

## Share the history of your commands

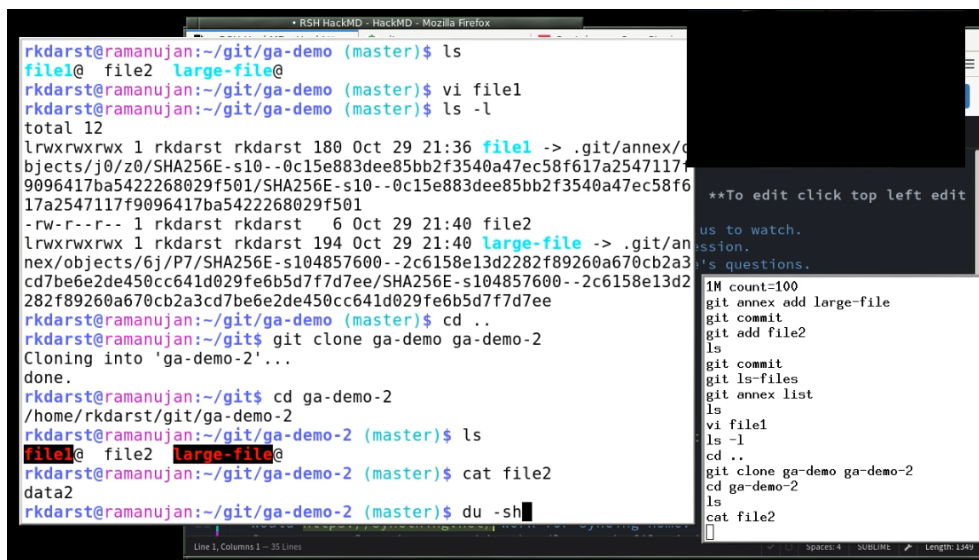
Even if you type slowly it is almost impossible to follow every command. We all get distracted or want to read up on something or while following a command fails on the learner laptop while instructor gets no error.

**Add pauses and share the commands that you have typed so that one can catch up.**

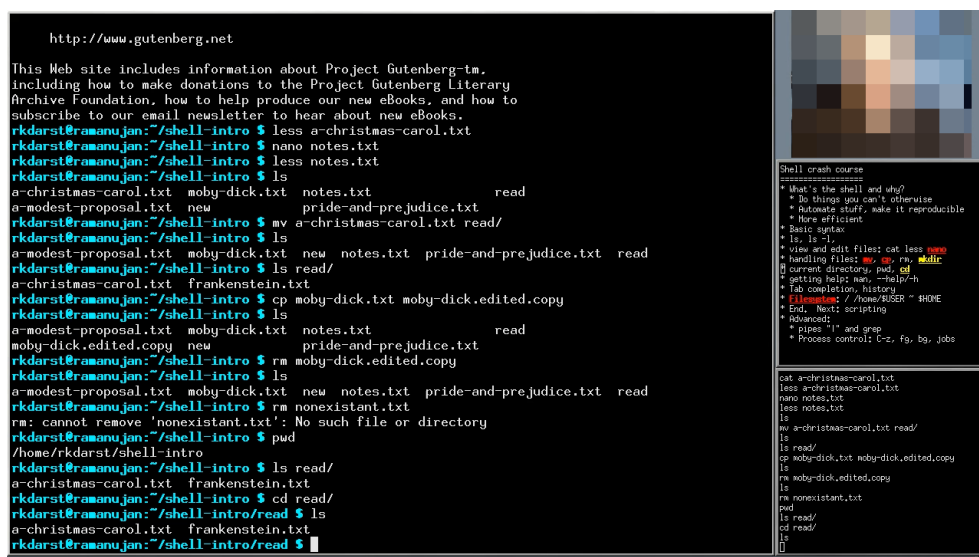
Below are some examples (some more successful than others) of sharing history of commands.



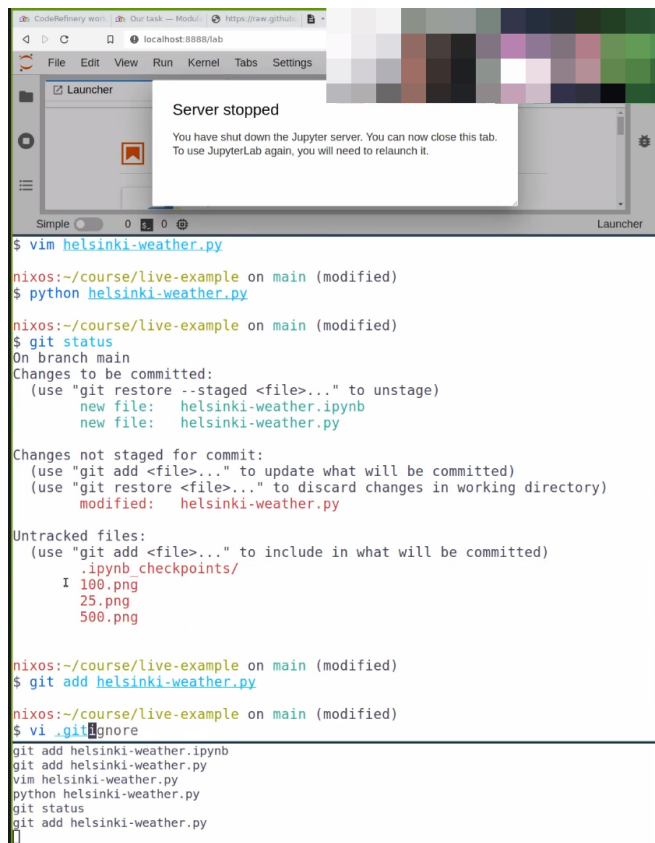
**H1:** A vertical screen layout shared. Note the extra shell history at the top. The web browser is at the bottom, because the Zoom toolbar can cover the bottom bit.



**H2:** This isn't a screen-share from CodeRefinery, but may be instructive. Note the horizontal layout and shell history at the bottom right.



**H3:** Similar to above, but dark. Includes contents on the right.



```

$ vim helsinki-weather.py
nixos:~/course/live-example on main (modified)
$ python helsinki-weather.py
nixos:~/course/live-example on main (modified)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   helsinki-weather.ipynb
    new file:   helsinki-weather.py

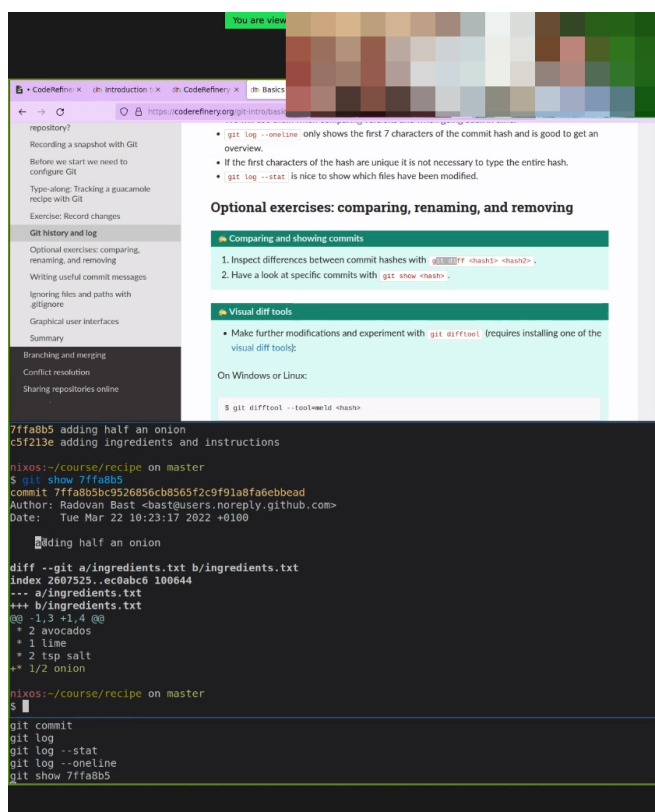
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   helsinki-weather.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .ipynb_checkpoints/
    100.png
    25.png
    500.png

nixos:~/course/live-example on main (modified)
$ git add helsinki-weather.py
nixos:~/course/live-example on main (modified)
$ vi .gitignore
git add helsinki-weather.ipynb
git add helsinki-weather.py
vim helsinki-weather.py
python helsinki-weather.py
git status
git add helsinki-weather.py

```

**H4:** Jupyter + terminal, including the `fish` shell and the terminal history.



```

7ffa8b5 adding half an onion
c5f219e adding ingredients and instructions
nixos:~/course/recipe on master
$ git show 7ffa8b5
commit 7ffa8b5bc9526856cb8565f2c9f91a8fa6ebbead
Author: Radovan Bast <bast@users.noreply.github.com>
Date:   Tue Mar 22 10:23:17 2022 +0100

    adding half an onion

diff --git a/ingredients.txt b/ingredients.txt
index 2607525..ec0abc6 100644
--- a/ingredients.txt
+++ b/ingredients.txt
@@ -1,3 +1,4 @@
 * 2 avocados
 * 1 lime
 * 2 tsp salt
+ * 1/2 onion

nixos:~/course/recipe on master
$ git commit
git log
git log --stat
git log --oneline
git show 7ffa8b5

```

**H5:** Lesson + terminal, `tmux` plus terminal history and dark background.



Array jobs — Aalto scientific.com

June 2022 / Intro / Array jobs

## Your first array job

Let's see a job array in action. Lets create a file called `array_example.sh` and write it as follows.

```
#!/bin/bash
#SBATCH --time=00:15:00
#SBATCH --mem=200M
#SBATCH --output=array_example_%A%.out
#SBATCH --array=0-15

# You may put the commands below:

# Job step
srun echo "I am array task number" $SLURM_ARRAY_TASK_ID
```

Submitting the job script to Slurm with `sbatch array_example.sh`, you will get

```
login3.triton.aalto.fi /scratch/work/darstr1/kickstart-2022
/scratch/work/darstr1
$ mkdir kickstart-2022
$ cd kickstart-2022/
$ nano array_example.sh
$ sbatch array_example.sh
Submitted batch job 5308576
$ slurm q
JOBID          PARTITION NAME          TIME          ST
(ON)
5308576_7      batch-csl array_example. 0:02 2022-06-
5308576_6      batch-csl array_example. 0:02 2022-06-
5308576_5      batch-csl array_example. 0:02 2022-06-
5308576_4      batch-csl array_example. 0:02 2022-06-
5308576_1      batch-csl array_example. 0:02 2022-06-
5308576_0      batch-csl array_example. 0:02 2022-06-
$ slurm q
JOBID          PARTITION NAME          TIME          ST
(ON)
$
```

```
cd $WRKDIR
pwd
mkdir kickstart-2022
cd kickstart-2022/
nano array_example.sh
sbatch array_example.s
slurm q
slurm q
```

**H6:** HPC Kickstart course. Note the colors contrast of the windows and colors of the prompt and text. The history is smaller and doesn't take up primary working space. The working directory is in the window title bar.

```
user@somewhere:~
$
```

**H7:** Show command history “picture-in-picture”, in the same terminal window.

## How to configure history sharing

You need to find a way to show the recent commands you have entered, so that learners can see the recent commands. Below are many solutions. Try them out and see what works for you.

- **prompt-log:** It adds a interesting idea that the command you enter is in color and also

provides terminal history before the command returns.

- **Simple:** The simple way is `PROMPT_COMMAND="history -a"` and then `tail -f -n0 ~/.bash_history`, but this doesn't capture ssh, sub-shells, and only shows the command after it is completed.
- **Better yet still simple:** Many Software Carpentry instructors use [this script](#), which sets the prompt, splits the terminal window using tmux and displays command history in the upper panel. Requirement: `tmux`
- **Better (bash):** This prints the output before the command is run, instead of after. Tail with `tail -f ~/demos.out`.

```
BASH_LOG=~/.demos.out
bash_log_commands () {
  # https://superuser.com/questions/175799
  [ -n "$COMP_LINE" ] && return # do nothing if completing
  [[ "$PROMPT_COMMAND" =~ "$BASH_COMMAND" ]] && return # don't cause a preexec
  for $PROMPT_COMMAND
  local this_command=`HISTTIMEFORMAT= history 1 | sed -e "s/^[ ]*[0-9]*[ ]*//"`;
  echo "$this_command" >> "$BASH_LOG"
}
trap 'bash_log_commands' DEBUG
```

- **Better (zsh):** This works like above, with zsh. Tail with `tail -f ~/demos.out`.

```
preexec() { echo $1 >> ~/demos.out }
```

- **Better (fish):** This works like above, but for fish. Tail with `tail -f ~/demos.out`.

```
function cmd_log --on-event fish_preexec ; echo "$argv" >> ~/demos.out ; end
```

- **Better (tmuxp):** This will save some typing. `TmuxP` is a Python program (`pip install tmuxp`) that gives you programmable `tmux` sessions. One configuration that works (in this case for `fish` shell):

```
session_name: demo
windows:
  - window_name: demo
    layout: main-horizontal
    options:
      main-pane-height: 7
    panes:
      - shell_command:
          - touch /tmp/demo.history
          - tail -f /tmp/demo.history
      - shell_command:
          - function cmd_log --on-event fish_preexec ; echo "$argv" >> /tmp/
demo.history ; end
```



- **Windows PowerShell:** In [Windows Terminal](#), a split can be made by pressing `CTRL+SHIFT+=`. Then, in one of the splits, the following PowerShell command will start tracking the shell history:

```
Get-Content (Get-PSReadlineOption).HistorySavePath -Wait
```

Unfortunately, this only shows commands after they have been executed.

- [Tavatar: shell history mirroring teaching tool](#) can copy recent history to a remote server.
- [history-window](#): Show command history “picture-in-picture” when teaching command line. Requires Bash.

## Font, colors, and prompt

### Terminal color schemes

- Dark text on light background, *not* dark theme. Research and our experience says that dark-text-on-light is better in some cases and similar in others.
- You might want to make the background light grey, to avoid over-saturating people’s eyes and provide some contrast to the pure white web browser. (this was an accessibility recommendation when looking for ideal color schemes)
- Do you have any yellows or reds in your prompt or program outputs? Adjust colors if possible.

### Font size

- Font should be large (a separate history terminal can have a smaller font).
- Be prepared to resize the terminal and font as needed. Find out the keyboard shortcuts to do this since you will need it.

### Prompt

At the beginning of the workshop your goal is to have a shell **as easy to follow as possible** and **as close to what learners will see on their screens**:

- Your prompt should be minimal: few distractions, and not take up many columns of text.
- [prompt-log](#) does this for you.
- The minimum to do is is `export PS1='\$ '`.
- Blank line between entries: `export PS1='\n\$ '`.
- Have a space after the `$` or `%` or whatever prompt character you use.
- Strongly consider the Bash shell. This is what most new people will use, and Bash will be

less confusing to them.

- Eliminate menu bars and any other decoration that uses valuable screen space.
- Add colors only if it simplifies the reading of the prompt.

Later in the workshop or in more advanced lessons:

- Using other shells and being more adventurous is OK - learners will know what is essential to the terminal and what is extra for your environment.

Try to find a good balance between:

- Showing a simple setup and showing a more realistic setup.
- Showing a consistent setup among all instructors and showing a variety of setups.

## Habits we need to un-learn

- **Do not clear the terminal.** Un-learn CTRL-L or `clear` if possible. More people will wonder what just got lost than are helped by seeing a blank screen. Push `ENTER` a few times instead to add some white space.
- **Do not rapidly switch between windows** or navigate quickly between multiple terminals, browser tabs, etc. This is useful during your own work when nobody is watching, but it is very hard to follow for learners.
- Avoid using **aliases** or **shortcuts** that are not part of the standard setup. Learners probably don't have them, and they will fail if they try to follow your typing. Consider even to rename corresponding files (`.bashrc`, `.gitconfig`, `.ssh/config`, `.ssh/authorized_keys`, `.conda/*`).
- Be careful about using **tab completion** or **reverse history search** if these haven't been introduced yet.

## Desktop environment and browser

- Try to remove window title bars if they take up lots of space without adding value to the learner.
- Can you easily resize your windows for adjusting during teaching?
- Does your web browser have a way to reduce its menu bars and other decoration size?
  - Firefox-based browsers: go to `about:config` and set `layout.css.devPixelsPerPx` to a value slightly smaller than one, like `0.75`. Be careful you don't set it too small or large since it might be hard to recover! When you set it to something smaller than 1, all window decorations become smaller, and you compensate by zooming in on the website more (you can set the default zoom to be greater than 100% to compensate). Overall, you get more information and less distraction.

## How to switch between teaching setup and work setup?

- Make a dedicated “demos” profile in your terminal emulator, if relevant. Or use a different terminal emulator just for demos.
  - Same idea for the browser: Consider using a different browser profile for teaching/demos.
  - Another idea is to containerize the setup for teaching. We might demonstrate this during the [Sharing teaching gems](#) session later.
- 

### 📌 Keypoints

- Share **portrait layout** instead of sharing entire screen
- **Adjust your prompt** to make commands easy to read
- **Readability** and beauty is important: adjust your setup for your audience
- **Share the history** of your commands
- Get set up a **few days in advance** and get feedback from someone else. Feedback and time to improve is very important to make things clear and accessible. 10 minutes before the session starts is typically too late.

## Exercises

### 👉 Evaluate screen captures (20 min)

Evaluate screenshots on this page. Discuss the trade-offs of each one. Which one do you prefer? Which are useful in each situation?

Please take notes in the collaborative document.

### 👉 Set up your own environment (20 min)

Set up your screen to teach something. Get some feedback from another learner or your exercise group.

## Other resources

- <https://coderefinery.github.io/manuals/instructor-tech-setup/>
- <https://coderefinery.github.io/manuals/instructor-tech-online/>