03. Installing Linux, Git, More Shell Commands

CPSC 120: Introduction to Programming
Kevin A. Wortman ~ CSU Fullerton

Agenda

0. Sign-in sheet
1. OSS Club Introduction
2. Q&A
3. Installing Linux
4. Git

1. OSS Club Introduction

2. Q&A

Q&A

Let’s hear your questions about...

- This week’s Lab
- Linux
- Any other issues

Reminder: write these questions in your notebook during lab

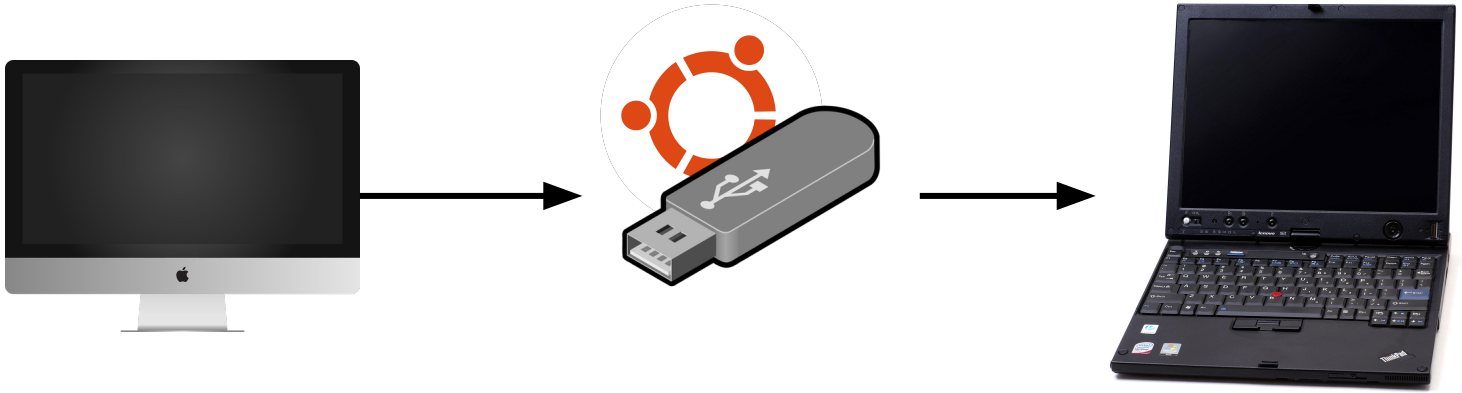
3. Installing Linux

Week 3 Lab

- Survey
- Installing Linux
- Bring your laptop
- USB drives are provided

Overview

- **Operating system**: software that manages hardware and provides platform for other software
  - macOS, Windows, Linux, ...
- Computer runs one operating system at a time
- Install Linux: copy Linux OS software to computer storage
  - Replaces existing OS



Download .iso

- .iso: “image” of contents of USB
- ubuntu-22.04.3-desktop-amd64.iso (5 GB)
- Use any computer to download
- Large
- USB must be 8 GB or larger (common)

Create USB

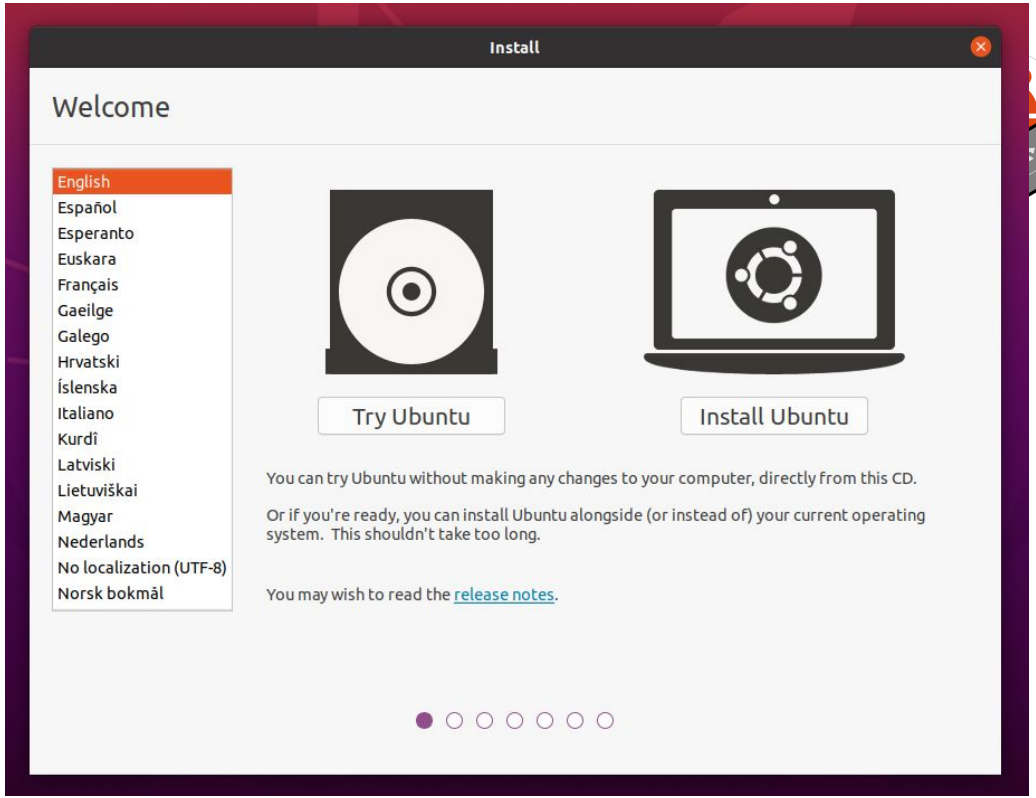
- Need to write ubuntu-22.04.3-desktop-amd64.iso to USB
- Erases USB contents
  - You can reformat after install
- Need to use image-writing software
  - balenaEtcher (macOS, Windows, Linux)
  - Startup Disk Creator (Ubuntu)

Boot from USB

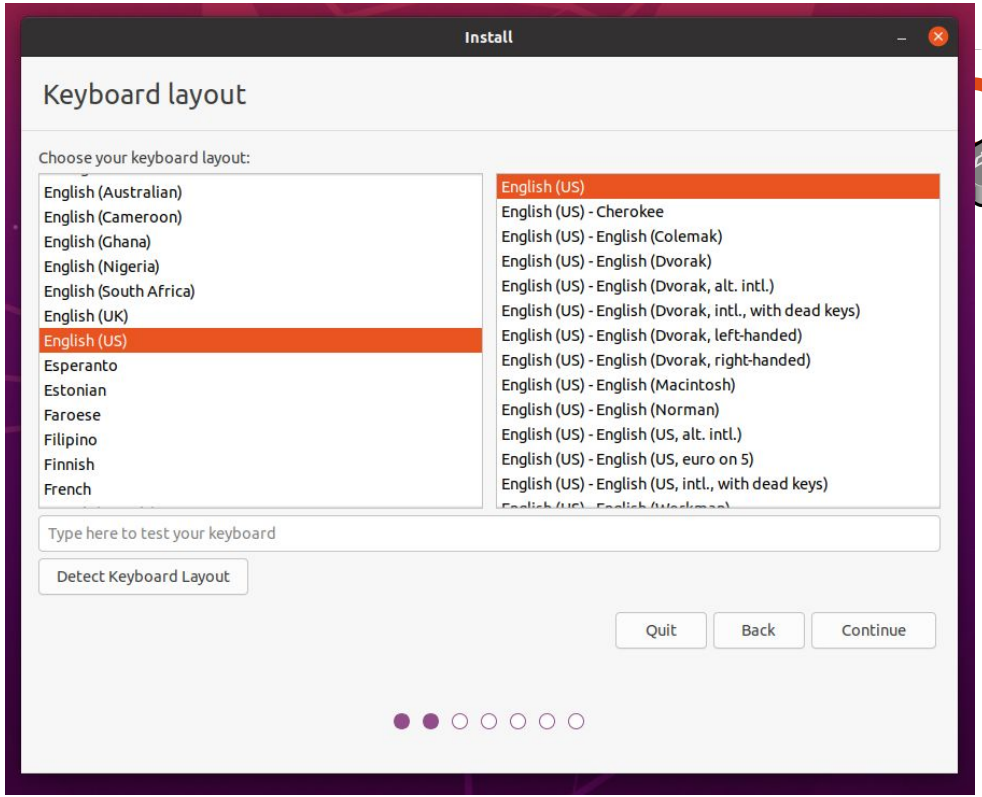
- Insert USB into computer for install
- Restart
- Wait for Power On Self Test (POST) = logo appears
- Press button for **Boot Menu**
- Possible boot menu keys:
  - **F12 (Lenovo, Dell)**
  - Escape
  - F2
  - F10
- In doubt: Google “*manufacturer* boot menu key”
  - Ex: “HP boot menu key”
  - (hardest part)

Boot Menu - Choose USB

Ubuntu Setup: Install

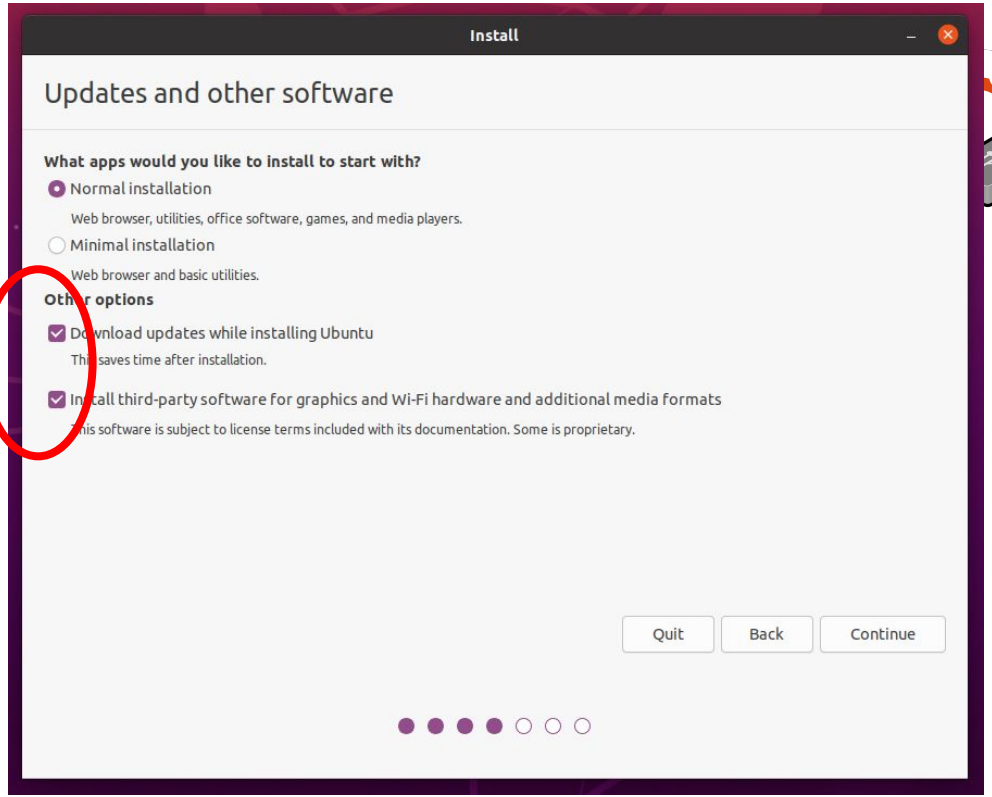


Ubuntu Setup: Language

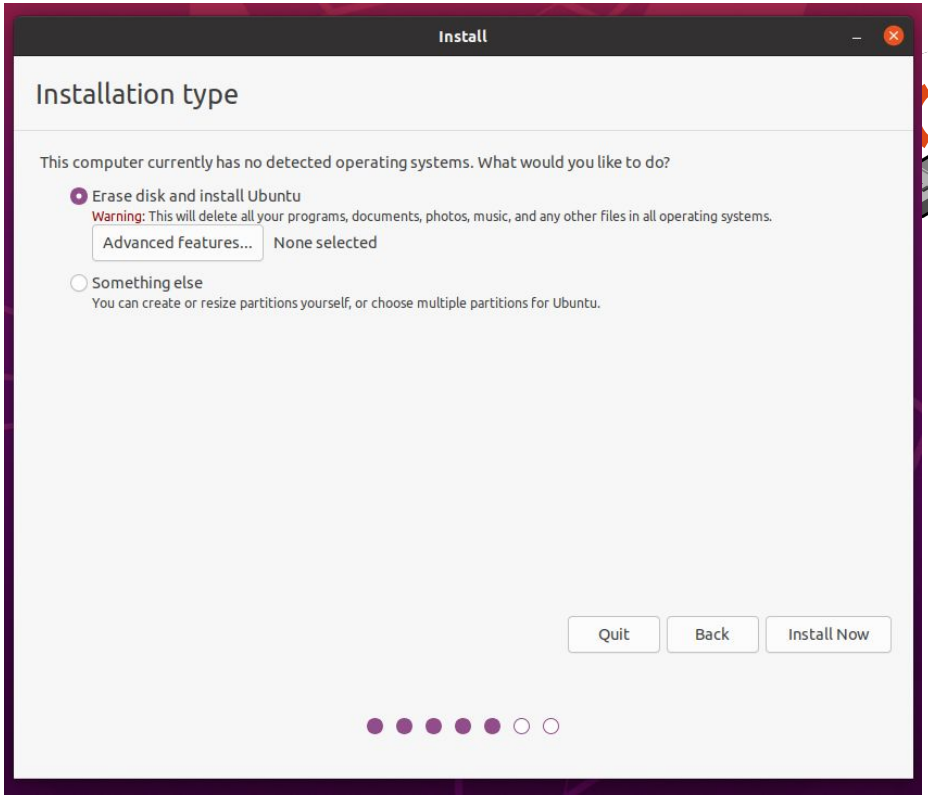


Ubuntu Setup: Updates

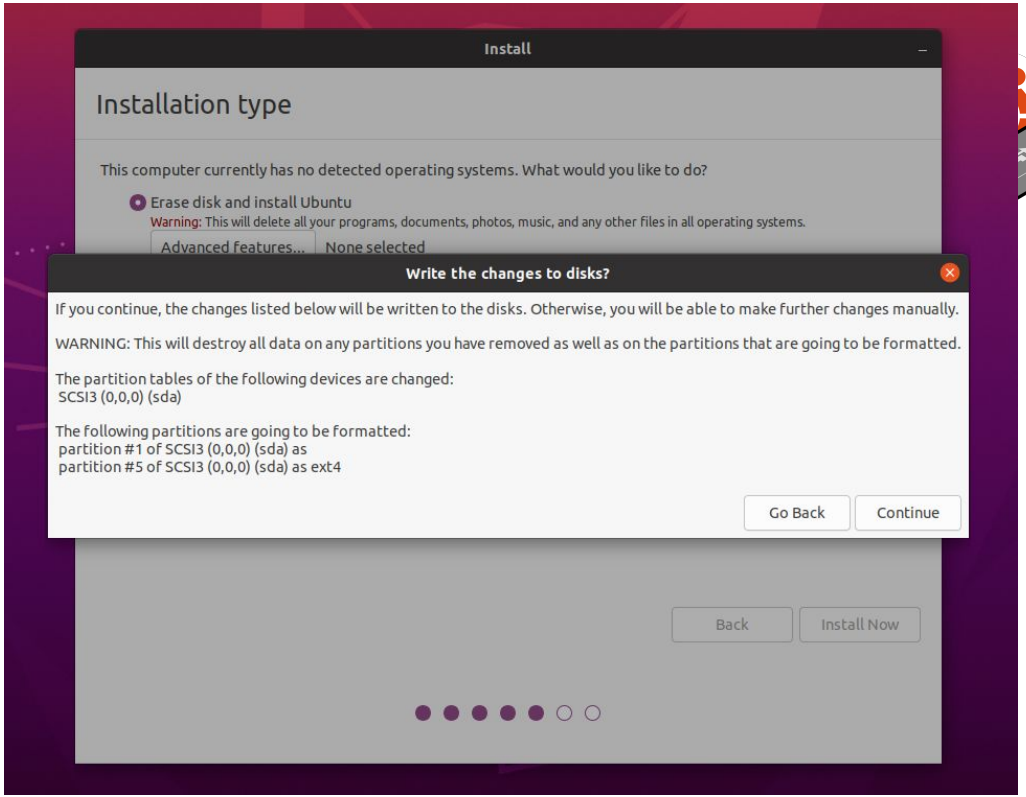
recommend
ticking these
(not essential)



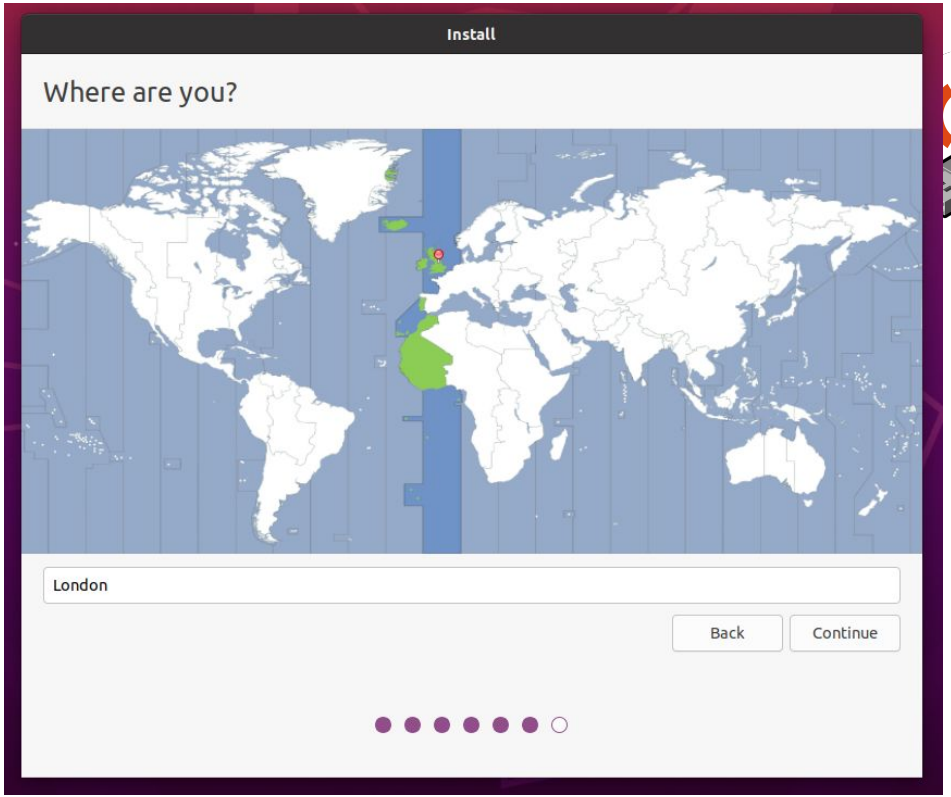
Ubuntu Setup: Disk



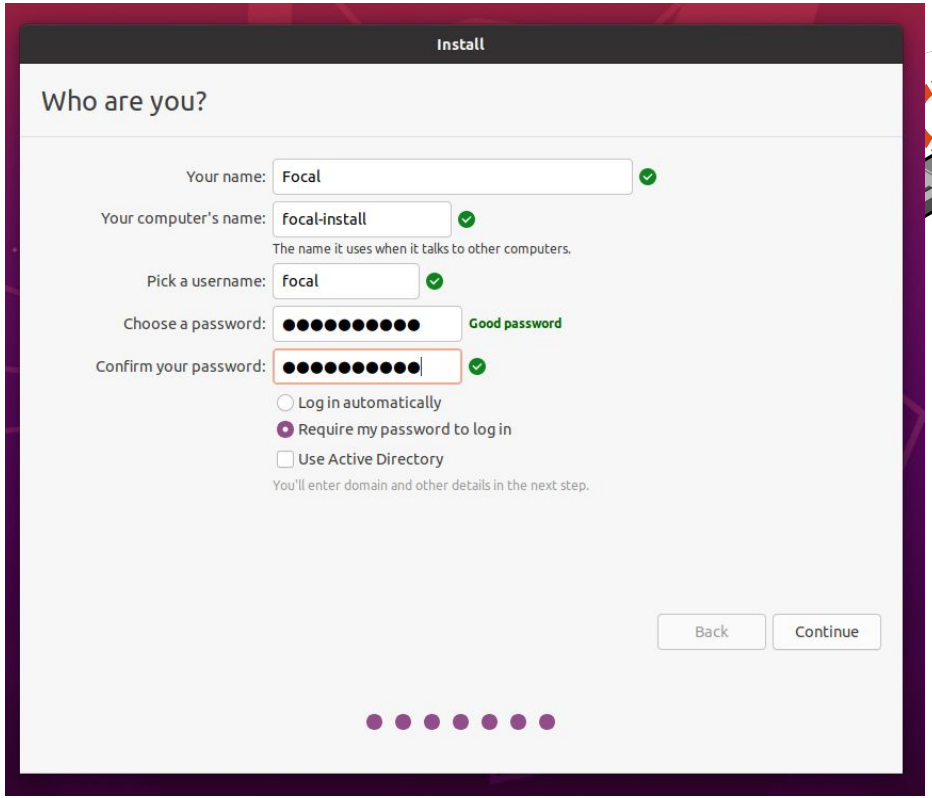
Ubuntu Setup: Confirm - Point of No Return



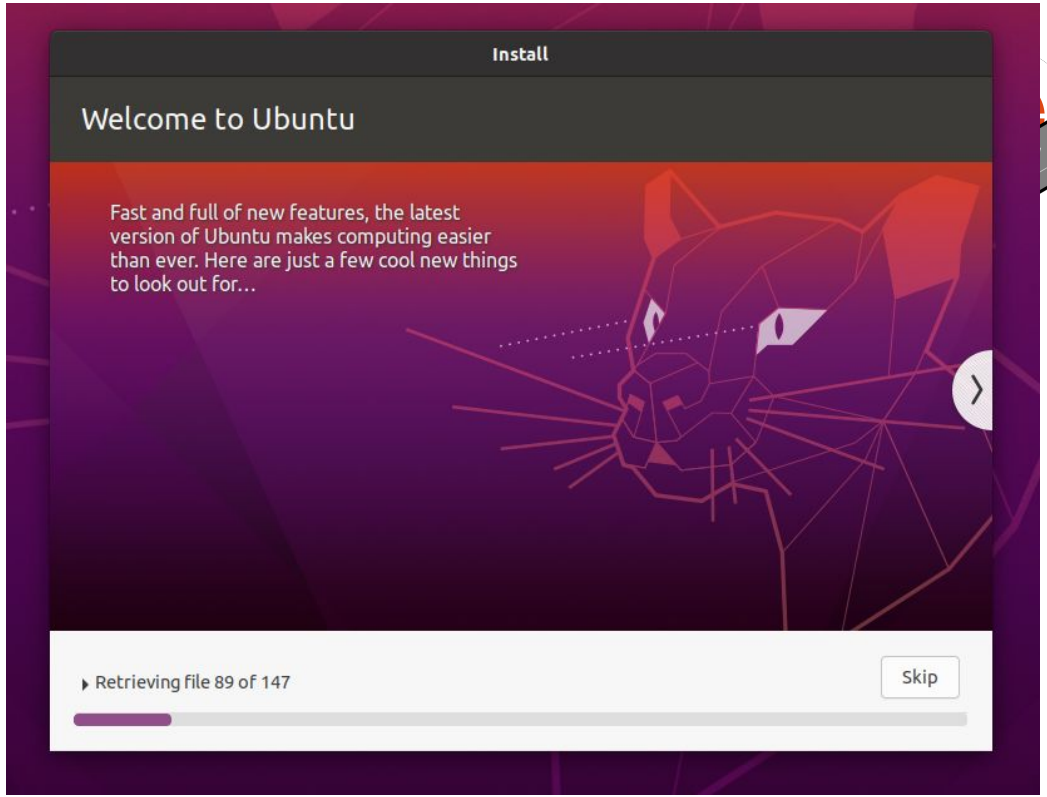
Ubuntu Setup: Location



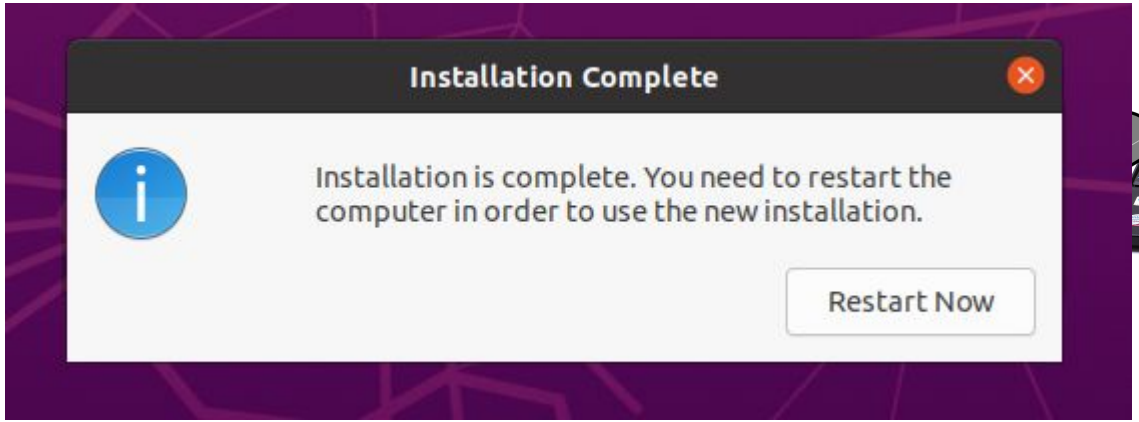
Ubuntu Setup: Username/Password



Ubuntu Setup: Copying

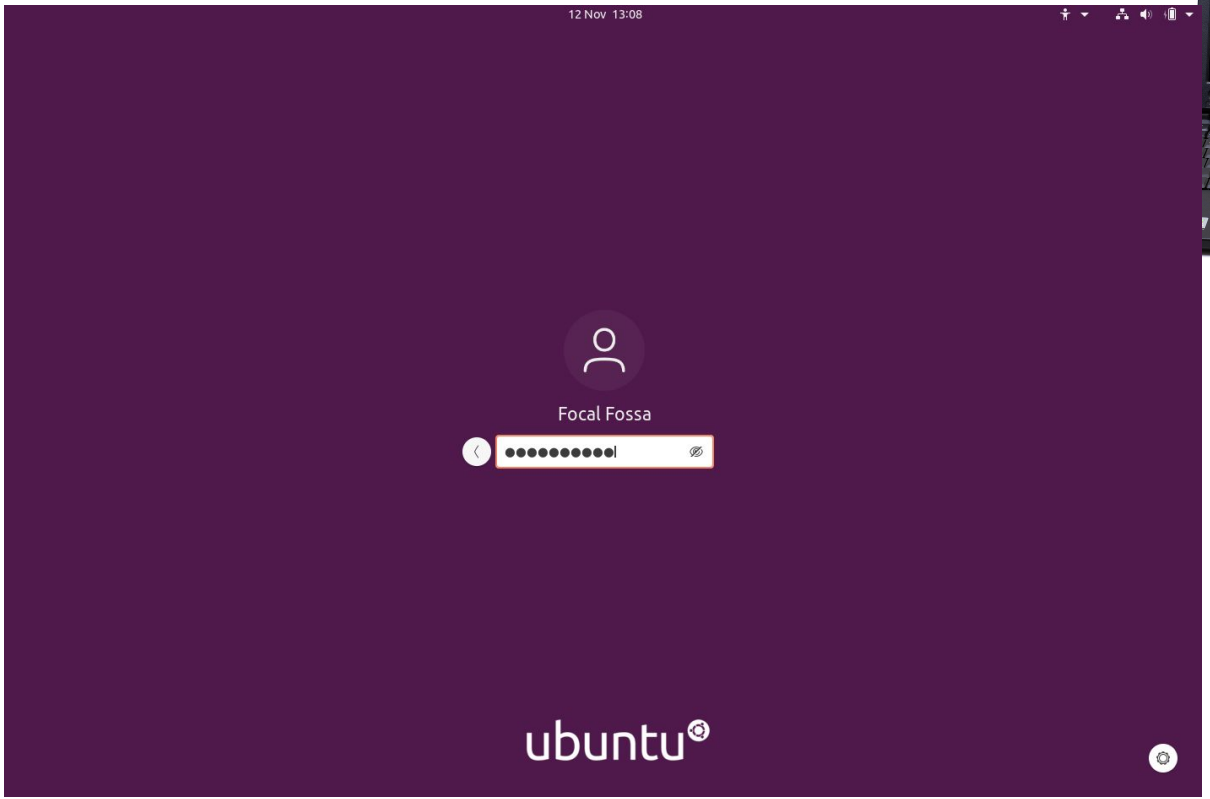


Ubuntu Setup: Reboot

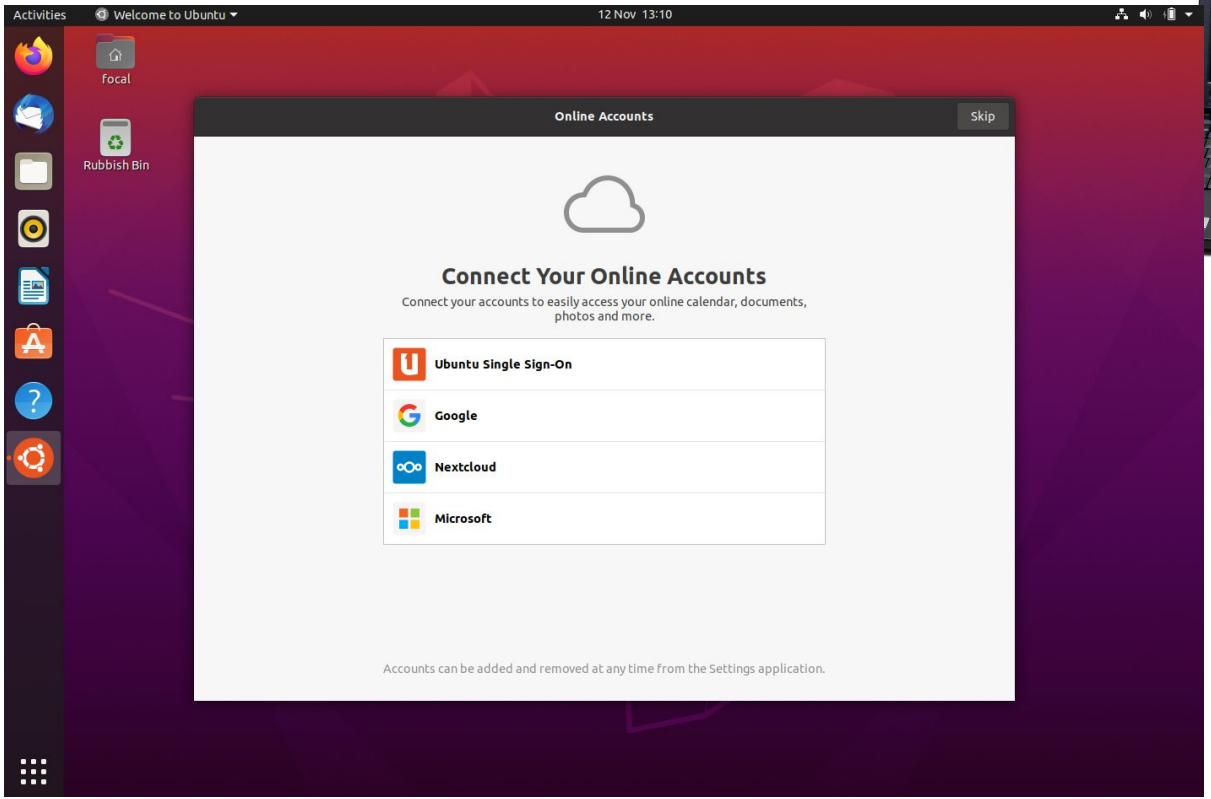


Remove the USB drive when it asks.

Ubuntu Setup: Login



Ubuntu Setup: Login



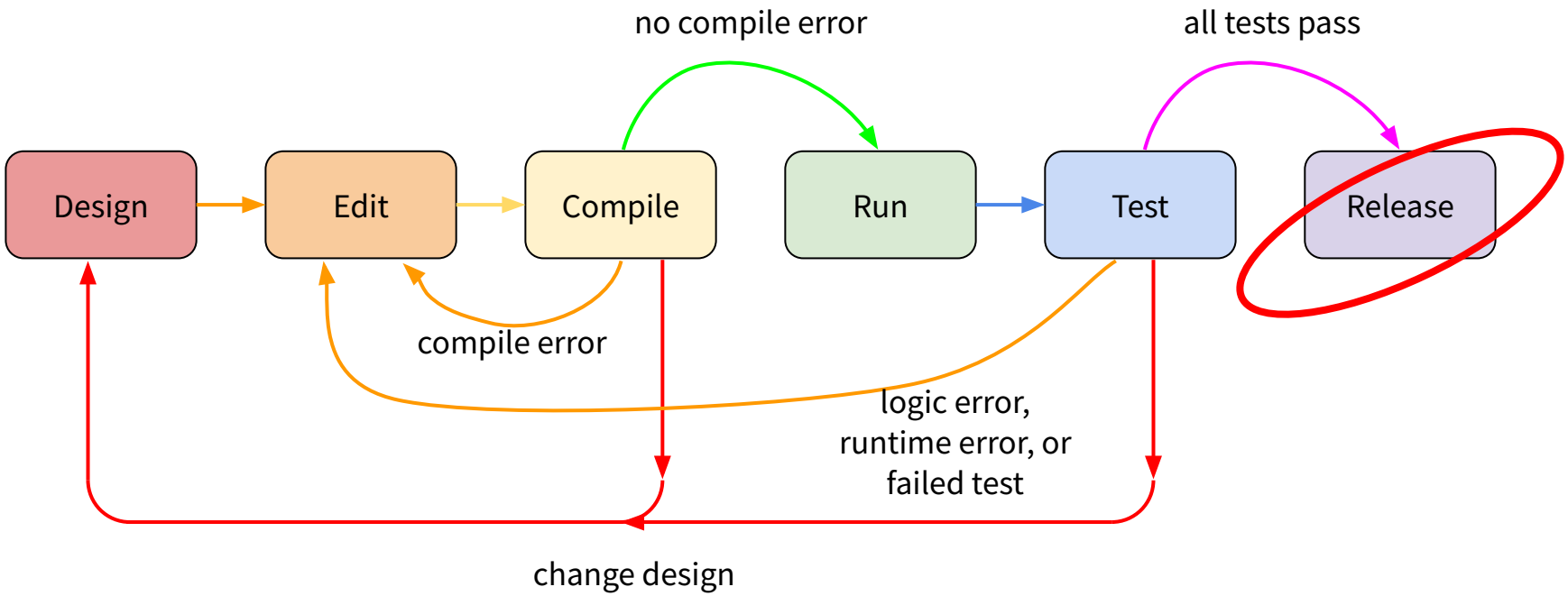
Development Tools (clang++, VS Code)

As described in Linux & Tools page in Canvas:

```
$ wget -q https://raw.githubusercontent.com/mshafae/tusk/main/quickinstall.sh -O- | sh
```

4. Git

The Development Cycle



Git and GitHub

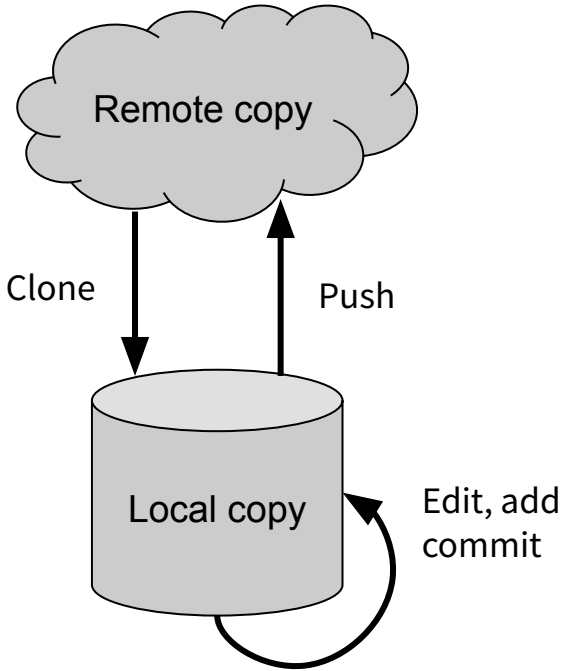
- **Source code control**: tool for programmers to share, track source code
- **git**: popular source code control shell program
- **GitHub**: cloud git service
  - facilitates sharing code with others around the world
- **Repository** (“**repo**”): holds a project
- Example: chromium, chrome_history_client.cc
- Our labs

GitHub Workflow

git understands that a repo can be copied into multiple places at the same time

single-developer workflow:

1. Create a **remote copy** repo (lives on github.com)
2. **Clone** a **local copy** onto your computer
3. Edit, save files inside local copy
4. Create **commit(s)** summarizing changes
5. **Push** commits to **remote copy**



Releasing Work to GitHub: Working Backwards

- Git records edits, who made them, when, why
  - see chrome_history_client.cc history
- **git push**: transmits every **commit** in your local repo to GitHub.com
  - First, a commit needs to exist
- **git commit**: logs a commit action
  - Applies to all currently-**staged** files
  - Commit **message**: human-readable text describing what you did
  - First, at least one file needs to be staged
- **git add**
  - **Stages** a file = “this file is part of the next commit”

Releasing Work to GitHub: Working Forwards

Working forwards…

1. Edit, save work in VS Code
2. Compile (`clang++`), run (`./a.out`), test
3. **Add**: for each FILE you changed,
   `$ git add FILE`
4. **Commit**: once,
   `$ git commit -m “MESSAGE”`
5. **Push**: once,
   `$ git push`
6. Check: reload repo in browser, confirm changes

git clone

```
$ git clone REPO
```

- REPO comes from the “Clone or download” button and ends in .git
  - For https://github.com/cpsc-pilot-fall-2022/hello-world
  - REPO-URL = https://github.com/cpsc-pilot-fall-2022/hello-world.git
- Download the contents of REPO into a directory on the local computer
- Prints status to stdout, even on success
- May ask for your GitHub username/password

git status

```
$ git status
```

- Must be run **inside a git repo**
- Prints out
  - List of files that have been modified, but not committed yet
  - List of all commits that haven't been pushed yet
- Quick way to check for un-pushed work

git add

```
$ git add FILE...
```

- Must be run **inside a git repo**
- Each of FILE… is hereby “**staged for commit**”
- (the next `git commit` will apply to them)

git commit

```
$ git commit -m “MESSAGE”
```

- Must be run **inside a git repo**
- Creates a **commit** that applies to all currently-staged files
- MESSAGE should be a human-readable description what the commit represents
  - E.g. “fixed the crash bug”, “finished lab 2”
- Note: **quotes** around MESSAGE !
- If you forget -m “MESSAGE” then git will open a venerable editor “nano” and force you to write a message that way

git push

```
$ git push
```

- Must be run **inside a git repo**
- Upload all local commits to the remote repo
- Synchronization check
  - git checks for commits that were pushed since you last cloned/pulled
  - If you are out of sync, git push fails
  - Have to git pull first
- Should make all local changes visible on github.com
- Best practice: after a git push, look at your repo in a browser, confirm it is up to date