CSE 168 Lab

- **Section 02L:**
  - TA: Adam Weingram
  - Kolligian (KOLLIG) 202, every Tuesday 4:30pm–7:20pm
- **Section 03L:**
  - TA: Yuke Li
  - Kolligian (KOLLIG) 202, every Thursday 4:30pm–7:20pm
- **Section 04L:**
  - TA: Adam Weingram
  - Kolligian (KOLLIG) 208, every Tuesday 7:30pm–10:20pm

- Linux system to complete labs.
- The computer labs on campus can actually boot to Linux by restarting and pressing F12 during the startup.
- Once you log off/shutdown the lab computer, the system is reset. Hence, try to save your files to Box/Drive to avoid losing your work.

What will lab sessions look like?

1. Short presentation from the TA on lab assignment details
2. TA answers questions about lab (those pertaining to all students)
3. Students work on the lab assignment, TA is available for individual questions and help with the lab

Note: Lab 0 will be a little bit different

Lab Expectations

- Written report
  - Typed, organized
  - Includes appropriate figures
  - Sources (if used) must be cited
- Code included
  - Code must be your own, unless project is explicitly group-based
  - You should be able to explain what your code is doing
- Code should actually run when submitted!
  - Don’t submit code with compiler errors, missing source files, etc.

Lab 0: Review of Linux, Containers

- Review the basics:
  - Using lab machines
  - Linux
  - Bash/Bash-like command line interface
- Docker Installation
- Build & Run a Docker image
  - *This is important; we'll use the resulting image as an environment for Lab 1!*

No report/submission necessary for Lab 0

Lab 1: RPC

- Latency benchmark
  - Measure the latency of a message between one RPC client and one RPC server.
  - Both client and server would run on the same machine (by running them on different Linux Shell or Terminal).
  - Note down either one-way latency or the round-trip time (RTT) of the message.
- Throughput benchmark
  - How many RPC calls per second can the server handle?
  - One server multiple clients.

Lab 2: Distributed Data Analysis

- Use Hadoop to process large amounts of data
- *More information on this later*

Lab 3: AI/ML Model Training

- **Group-based** project
- Build, train, and apply a machine learning model
- More open-ended; be creative!
  - *Feel free to start thinking of ideas!*
- You can use **Tensorflow** or **PyTorch**, but provided materials will assume Tensorflow

The course

What are you all hoping to get out of the CSE 168?

Your Experience

Reporting **not** having experience with:

Linux	27 respondents	**33 %**	
Docker	59 respondents	72 %	
Git	25 respondents	30 %	
Vim	58 respondents	71 %	
C	14 respondents	17 %	
C++	6 respondents	7 %	
Java	16 respondents	20 %	
Python	8 respondents	10 %	
No Answer	3 respondents	4 %	

From the "Warm Up" survey assigned on August 30th

Lab 0

Review of Linux and Containers

Linux

- An open-source operating system originally created by the Finnish software engineer Linus Torvalds
  - The core of which is the Linux **kernel**
  - Fun fact: Torvalds also created git after becoming frustrated with other version control software while working on the Linux kernel
- Used *everywhere* today; from embedded to phones to servers
- We will be using Ubuntu, a distribution of Linux for our labs

To boot into Linux on your lab machine, please restart it while holding the F12 key

The Command Line

- If you're a CSE major, become familiar with command line interfaces sooner rather than later (especially on Linux)!
- **If you are new to Linux, please become familiar with basic Linux commands by following:** https://ubuntu.com/tutorials/command-line-for-beginners#1-overview.
- Need help? Manual pages have lots of information!
  - Try: `man example_command`
  - Press "q" to exit
- *Those of you with experience, please help those around you!*

Shell/Bash Scripting

- We can write scripts to automate repetitive tasks
- Similar to programming languages you may be familiar with
- A basic shell script is often a list of commands that should be executed in sequence
- Use an editor like gedit, nano, or vim to create and edit a file with a ".sh" extension
- **Try to write a script that:**
  - **Uses a "shebang" to tell the machine it's a bash script (**#!/usr/bin/env bash**)**
  - **Creates a directory (**mkdir my_dir**)**
  - **Adds a new file to that directory (**touch ./my_dir/my_file.txt**)**
  - **Downloads a web resource (**wget https://www.gnu.org/software/wget/manual/wget.txt**)**
  - **Renames the directory (**mv my_dir my_folder**)**
  - **Has executable permissions (**chmod +x ./my_script.sh**)**
- Those of you who are more comfortable, use a for loop or find+xargs to append some text to every file in the directory too!

Containers

- Linux also provides some nice features that we can use to create **containers**
- Somewhat like a virtual machine, but with less overhead
- The kernel is shared, but programs in different containers are isolated† from one another and from programs running on the host
- We will be using **Docker** for our labs
  - Container ecosystem, management, and (in some cases) VM
  - Will let us create consistent and organized development environments with fewer dependency management issues

† To varying degrees, depending on the implementation

Install Docker

Lab Machines:

https://docs.docker.com/engine/install/ubuntu

Personal Machines:

https://docs.docker.com/engine/install

Writing a Dockerfile

- Kind of like a shell script, but describes a Docker “Image”
- Some important instructions:
  - `FROM` :: Create a new build stage and define a base image
    - e.g., `FROM ubuntu:18.04`
  - `RUN` :: Run a command
    - e.g., `RUN cd hello-world`
  - `ENV` :: Set an environment variable
    - e.g., `ENV VERSION_NUMBER=1.3`
  - `COPY` :: Copy files from the local system into the container
    - e.g., `COPY my_code.cpp /project/my_code.cpp`

Your Task

- Look at Lab 1 instructions
- **Create a Docker image with all of gRPC's dependencies installed**

See: https://grpc.io/docs/languages/cpp/quickstart

Lab 1

Remote Procedure Calls (RPC)

RPC setup

- Before diving into the Lab 1, we need to setup our Linux environment to use/run RPC framework.
- Out of the three popular RPC systems we will use gRPC in C++ for the lab. But you can use Apache Thrift too.
- Steps to install gRPC can be found here- https://grpc.io/docs/languages/cpp/quickstart/ (Please follow the steps carefully).
- To confirm your installation went correctly, you can test it out by running a simple client-server application with gRPC- https://www.grpc.io/docs/languages/cpp/quickstart/#build-the-example
- Detailed instructions about lab evaluation will be shared soon.