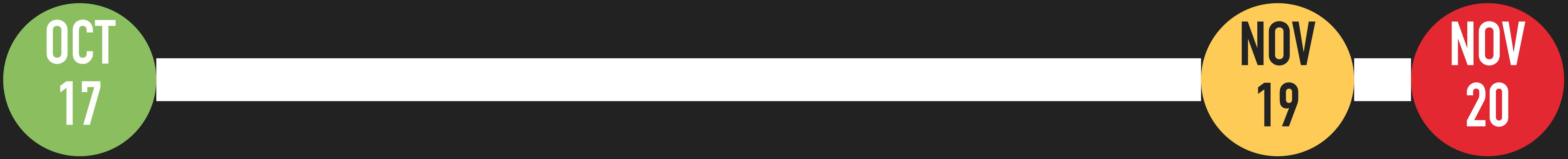
SOFTWARE-DEFINED SYSTEMS FINAL PROJECT



2110415 SEMESTER 1/2023

Kunwadee Sripanidkulchai, Ph.D.

pi kube cluster in a box

CHULA ΣNGINEERING Foundation toward Innovation COMPUTER

Ingredients

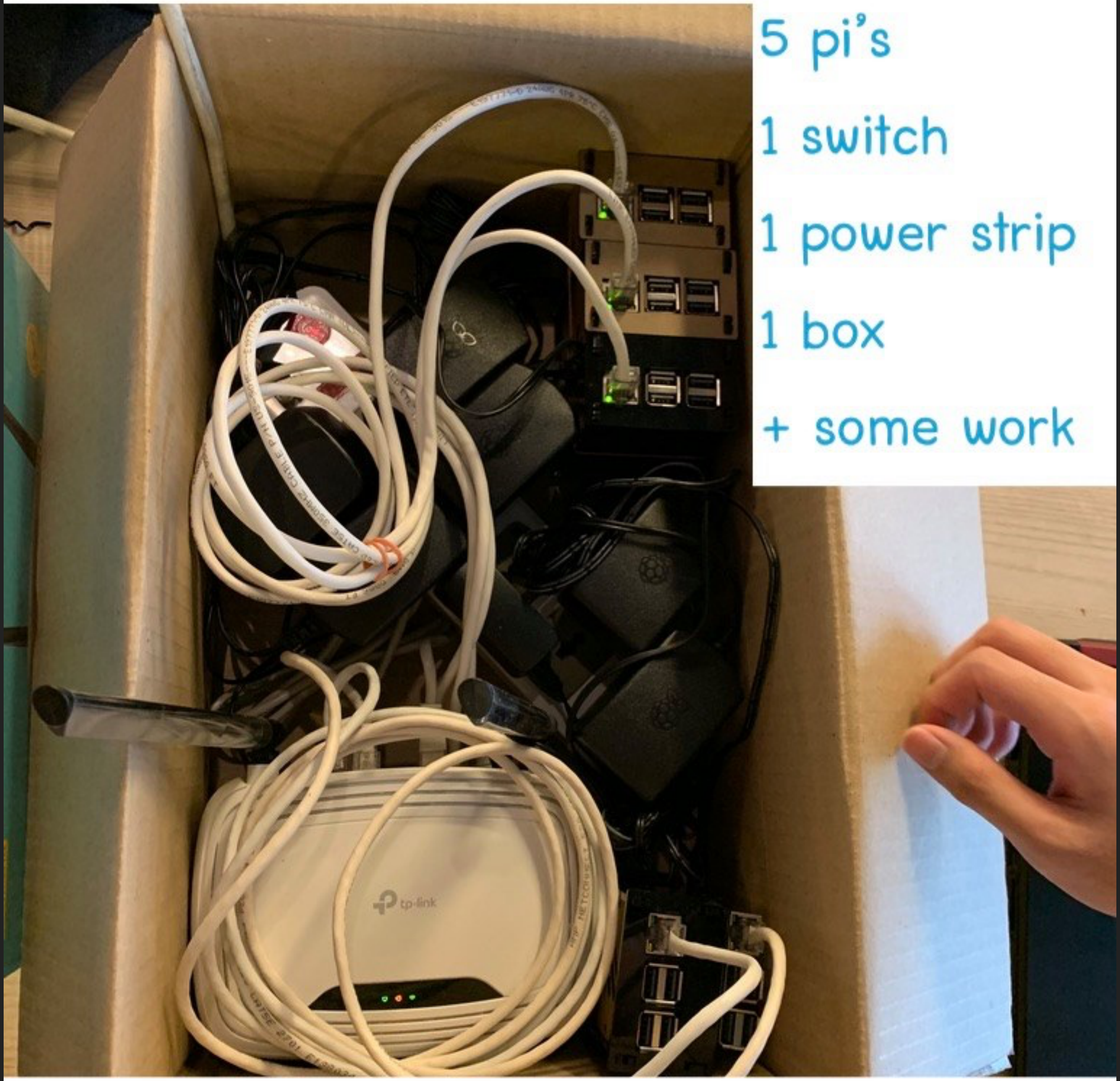
5 pi's

1 switch

1 power strip

1 box

+ some work



2110415 Software-Defined Systems
offered every first semester

DEVICE CHECKLIST (PER GROUP)

- 1 x Ethernet/WiFi Router TP- Link TL WR841N
- 4 x Raspberry Pi Kit
  - Raspberry Pi 3 B+
  - Power Adapter
  - SD Card
  - Ethernet Cable
  - Ethernet Adapter and USB-C/USB-A Adapter as needed

TEST YOUR PI

1. Turn on your router
2. Connect Pi's to LAN
3. Confirm that your Pi's can use the network (ping, ssh)

OBJECTIVES

- Build a Kuberbetes cluster using at least 4 Raspberry Pi's as nodes (the minimum number of Pi's is equal to the number of members in your group) and at least 2 of your own notebooks as Masters (controllers). You may choose to use wired or wireless connectivity.
- Find/build an application to deploy on your cluster. Your application must use at least 4-5 **different** containers providing **different services** (the minimum number of containers is equal to the number of members in your group).   You may choose an application with any architecture: monolithic or microservice.
- The containers must be dependent on each other, meaning that a client request must be serviced by **at least 2 containers** before a response is sent back to the client. This requires networking containers on different nodes (yet another application of SDN).
- **Your application must be fault-tolerant** in that if any single node (Raspberry Pi) and its containers go offline, your system will generate replacements and your application will automatically continue to work after the replacements are up and running. (The app can fail until the point where the replacements are running).

APPLICATION CONSIDERATIONS

- If you chose a microservice for your application, the way you break up your services needs to "make sense", i.e., be logical and practical. Do not break things up just for the sake of this project.
- A stateless application is the easiest choice for this project. A stateful application will require more effort. Think about why and chose your application wisely.
- You may chose to implement a simple stateless application, as long as the application logic and services make sense.

WHAT TO SUBMIT IN MCV BY NOVEMBER 19, 2023 23:00 GITHUB REPO

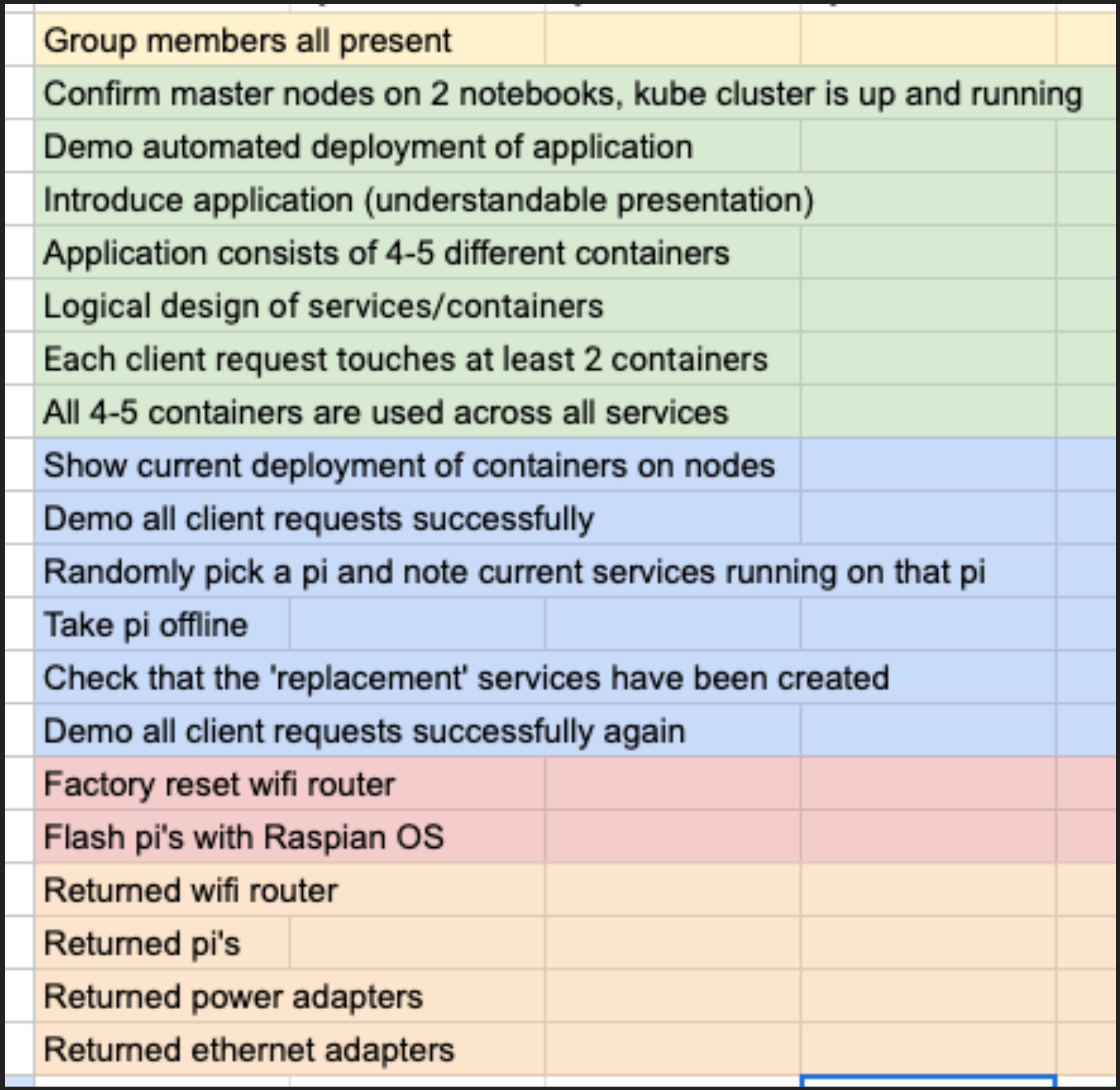
- Private github repo with README for
  - how to set up your Kubernetes cluster
  - how to deploy your application
- Following your instructions, I should be able to create my own cluster and deploy your chosen application on it.
- Give me (kunwadee) and the TA (theminer3746) access to your repo.

WHAT TO DEMO IN CLASS 9AM-12PM NOVEMBER 20, 2023

- Demo automatic deployment of your application with all components available in your Github repo
- Demo your application working with successful client requests, and demo that each client request is serviced by 2 or more containers
- Demo fault tolerance in your application by pulling each Raspberry Pi offline, one at a time. Client requests should be successful after containers are online.

STEPS/RUBRIC FOR DEMO IN CLASS 9AM-12PM NOV 20, 2023

- Have your demo set up and ready before your the start of your demo time slot
- Demo automatic deployment of your application with all components available in your Github repo
- Demo your application working with successful client requests, and demo that each client request is serviced by 2 or more containers
- Demo fault tolerance in your application by pulling a Raspberry Pi offline (instructor will pick a random one). Client requests should be successful after containers are online.
- After demo, sign in and return all equipment
  - Flash all Raspberry Pi's with default Raspian OS
  - Factory reset wireless router



Group members all present
Confirm master nodes on 2 notebooks, kube cluster is up and running
Demo automated deployment of application
Introduce application (understandable presentation)
Application consists of 4-5 different containers
Logical design of services/containers
Each client request touches at least 2 containers
All 4-5 containers are used across all services
Show current deployment of containers on nodes
Demo all client requests successfully
Randomly pick a pi and note current services running on that pi
Take pi offline
Check that the 'replacement' services have been created
Demo all client requests successfully again
Factory reset wifi router
Flash pi's with Raspian OS
Returned wifi router
Returned pi's
Returned power adapters
Returned ethernet adapters

HELPFUL HINTS

- Use Ubuntu VM's (at least 2 CPU cores) on your notebooks as Masters.
- Make your very own Kubernetes cluster from Raspberry PI
  - https://medium.com/nycdev/k8s-on-pi-9cc14843d43
  - https://ubuntu.com/tutorials/how-to-kubernetes-cluster-on-raspberry-pi#1-overview
  - This links are a good starting point. Not every thing will work.