COMP S381F Lab 01 Server-Side Development Tools

Lab Tasks & Questions (50~60 mins)

This tutorial includes 4 parts of lab tasks (**A,B,C,D**) and 8 questions (**Q1~8**). Your instructor will explain and do a demo for all lab tasks (except for `after-class` tasks). Please come early and try to complete all lab tasks with your instructor.

Should you have any questions about tutorial materials, please contact Dr. Alin Liu (ylliu@hkmu.edu.hk).

A. Warm-ups (10 mins)

Review the lecture notes or google to answer the below questions. Different answers are also acceptable as long as you have strong justifications.

Q1. What is client-server networking model? Briefly describe the differences between a server and a client.

Q2. What are Server-side apps? What are the server-side programs used to do?

Q3. What are cloud and cloud computing? What are cloud server apps?

B. Use Development Tools (15 mins)

VM VirtualBox (or VMware Workstation player/Pro) is a platform for running a single virtual machine on a Windows or Linux. We will use the virtual machine created by it to develop the server app, so any running crash or bugs caused won't affect the computer system.

Ubuntu is a Debian-based Linux operating system that runs from the desktop to the cloud, to all your internet-connected things. Our development will use Ubuntu to write and run project files.

References:

- Oracle VM VirtualBox Overview

  "https://www.oracle.com/assets/oracle-vm-virtualbox-overview-2981353.pdf"

- Introduction of VMware Workstation Player

  "https://www.vmware.com/info/workstation-player/evaluation"

- Introduction of Ubuntu

"https://ubuntu.com/blog/ubuntu-pro-24-04-lts-lands-on-google-cloud-power-up-your-cloud-experience"

- Is it recommended for developers to work on local virtual machines for development?

  "https://stackoverflow.com/questions/744152/is-it-recommended-for-developers-to-work-on-local-virtual-machines-for-developme"

1. VMware Workstation Player and Ubuntu (5 mins)

Notes:

- 4 JCC lab rooms installed the software required for this course: `D0625, D0626, D0627, and D0628`.
- To use computers in lab rooms, teaching staff and all student should login via id `guest0701` and password `guser1024`.

Entering Ubuntu in JCC lab computers via the following steps:

(a) Login the computers

(b) Boot into the `EWin10` partition

(c) Run VM VirtualBox

(d) Play virtual machine - `Ubuntu-64-22-04-4-Its`. The login/developer account and password for this Ubuntu are `student` and `student`, respectively.

Q4. Why do we use VM VirtualBox for development?

2. Run Ubuntu commands (10 mins)

References:

- 25 basic Ubuntu commands with detailed descriptions, syntax, examples, and explanations

  "https://www.geeksforgeeks.org/25-basic-ubuntu-commands/"

- The Linux command line for beginners

  "https://ubuntu.com/tutorials/command-line-for-beginners#1-overview"

Open a terminal in Ubuntu. Type the following Ubuntu commands for exercises.

(a) Access the **home** directory: `cd ~`

(b) Show the current path: `pwd`

(c) List all files in current folder: `ls`

(d) Create a folder: `mkdir` + the folder name

(e) Access a folder: `cd` + (the path/)the folder's name

(f) Create a code file: `gedit` + the file name

`E.g., `gedit server.js`.`

(g) Remove a file or a folder: `rm -rf`+ file's name/folder's name.

(h) Open a code file: `gedit` + the file name

`E.g., `gedit server.js`.`

Q5. Google and find what is the command `gedit` used for in Ubuntu?

3. Install VMware Workstation or VirtualBox (after-class)

This task guides you to download and install VM and Ubuntu on your own PC, which is not part of OCAS. Try your best to finish them. It's ok if you aren't familiar with them now. *You can use the school lab rooms (when they are available) for your after-class practicals or exercises.*

4 JCC lab rooms installed VM VirtualBox and Ubuntu are:

- D0625, D0626, D0627, D0628

(a) Download **the same Ubuntu image** used in lab rooms.

- Method 1: Copy **the Ubuntu image** in lab computer (check in `D0625, D0626, D0627, and D0628 > EWin10`)
  - "E:\virtual_machine\virtual_box\ubuntu_64_bit_22_04_4_LTS\ubuntu-64-22-04-4-lts" (This image is only used in `VM VirtualBox`.)
- Method 2: Download **the Ubuntu image** from the google drive link
  - "https://drive.google.com/file/d/1Qd_feLul52KwVyb_HTSMY55fjoPykQz5/view?usp=sharing" (This image is only used in `VM VirtualBox`.)
  - "https://drive.google.com/file/d/1z8rAe4J43Fr81hqEkrcUioHAL88Up4N7/view?usp=sharing" (This image is only used in `VMware Workstation Player`.)

(b) Download and install VM VirtualBox or VMware WorkStation Player (depend on your need)

- Download and install VM VirtualBox

  "https://www.virtualbox.org/wiki/Downloads"

- Download and install VMware WorkStation Player

  "www.vmware.com/asean/products/workstation-player/workstation-player-evaluation.html"

(c) Copy and run the Ubuntu image in your PC

- Running Ubuntu Using VM VirtualBox

  "https://www.youtube.com/watch?v=x5MhydijWmc"

- Running Ubuntu Using VMWare WorkStation Player

  "https://www.youtube.com/watch?v=SgfrHKg81Qc"

C. Run a sample app on Ubuntu (20 mins)

Study and run your first server app (written by Node.js). This app will run a server that provides network services (i.e., an HTTP response shows the string "Hello World!") to the client.

You will learn the complete running procedure of a Node.js-based server app. You will see the general project file formats, scripts, and running commands, e.g., Node.js scripts, package.json scripts, and npm commands. More details of them will be introduced in future lectures.

1. Download the app files from GitHub

All sample codes of this course will be included in my GitHub repository:

`https://github.com/yalin-liu/cloudserver381-2024/`

They can be downloaded in Ubuntu by running the following command:

```
git clone https://github.com/yalin-liu/cloudserver381-2024.git
```

Notes:

- `git clone` will download all files in one repository.
- If you only want download the codes from 1 particular lab folder of the repository, e.g., `https://github.com/yalin-liu/cloudserver381-2024/tree/main/lab01`, copy the link to [DownGit].
- This is very important for you to download the file on your PC. Because my repository files could be updated from time to time (e.g., updating the sample codes for a particular lab before it starts), you need to update the files for the lab.

(a) Open a terminal in Ubuntu, access the home directory and download the sample app to your *home* directory.

```
cd ~
git clone https://github.com/yalin-liu/cloudserver381-2024.git
```

(b) Enter the folder `cloudserver381-2024/lab01/helloworld`.

```
cd ~/cloudserver381-2024/lab01/helloworld
```

(c) Use the Ubuntu command `gedit` to open the project files and check the content.

```
gedit server.js
gedit package.json
```

Notes:

- `server.js` uses `express` (a Node.js framework/package/library) to create the server app.
  - The app can listen to the port number `8099`. Adding to the local host address of the server, the app will use `localhost:8099` to listen to the network requests from clients.
  - Once the app accepted the network request `get` in the url path `/`, it will response a string `Hello World!`.
- `console.log` is generated for local records of the server.
- `package.json` gives some basic information about the app, including `name`, `description`, `author`, and `version`. In addition, the `dependencies` field states the required node modules to be installed and support the app. The `engine` field states the version of `Node.js` to run the app. The `scripts` field packages all required Ubuntu commands to run the app. Once it works, all Ubuntu commands in the `scripts` field will be executed by only typing `npm start`.

Q6. In `server.js`, what network services it provides. In `package.json`, what it provides for the server app.

2. Run the server app `helloworld`

(a) Enter the folder `cloudserver381-2024/lab01/helloworld`.

```
cd ~/cloudserver381-2024/lab01/helloworld
```

(b) Run the server app using `npm`. Below two commands are for installing the server app's dependencies and run the server app in local machine.

```
npm install
npm start
```

Notes:

- Introduction of `npm`.

  "https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager"

- `npm install` - Install packages based on dependencies in the `package.json` file and auto create the `node_modules` folder and `package-lock.json` file
- `npm start` - It will start running the `server.js` file and create a server on localhost listening on port 8099
- `node_modules` and package-lock.json - It comes from the package.json file. When trying to execute "npm install" command

(c) When the server is running successfully, you will see the below `console.log` in the terminal.

```
Example app listening at http://localhost:8099
```

(d) Test the server app `helloworld`.

- Keep the terminal running the server.
- Open a browser (assuming you are a client user) and enter `http://localhost:8099` to send a `http get` request to the server (that is listening at `localhost:8099`).
- If things go well, your browser will get the response `Hello World!`.

Q7. What is the response if entering `http://localhost:8099/index` from the browser? Describe the response and explain why.

(e) Stop the server app.

- Method 1: Enter the terminal that runs the server, type `Ctrl + C`.
- Method 2: Stop the terminal (or delete the window).

D. Practicals (5~15 mins)

The server app `express-weather` responds to an HTTP web and returns the weather in global regions. Your instructor will do a quick demo for you. If you don't have time to complete this task in class, please do it as an after-class exercise.

1. Check the server app `express-weather`

(a) Go into the folder containing the `express-weather` app.

```
cd ~/cloudserver381-2024/lab01/express-weather
```

(b) Use the Ubuntu command `gedit` to open the project files and check the content.

```
gedit server.js
gedit package.json
```

Notes:

- In `server.js`, node-fetch is used for network services. It provides a lightweight module for the fetch API. The usage in the program is to obtain the weather API provided by the openweathermap website.

2. Add `API` to `server.js` (Allow your server accessing the outside database)

(a) Create your OpenWeather account. The `express-weather` app needs to access the outside database (i.e., api.openweathermap.org) through the **API** key.

To do this, you need to use your email to creat a free account from openweather and obtain your **API** key.

- Prepare **your email address** (e.g., O365 student email) for later use.
- Create a free account at [Openweather]. Write down your **login email** and **password** for later use.
- Each OpenWeather account can be allocated with an `API` key. Writ down your **API** key for later use.

(b) Add the **API** key to the `server.js` file.

- Use `gedit` to open the `server.js` file.
- Insert the **API** key into the double brackets of the const `APIKEY`.
- Save and exit the `server.js` file.

3. Run the server app `express-weather`

(a) Install the server app's dependencies and run the server app in local machine..

```
npm install
npm start
```

(b) Test your app by opening `http://localhost:8099` in your web browser. If things go well, you can *submit* the weather query for a region, e.g., Tokyo, and then see the following message.

Q8. What is the response if entering `http://localhost:8099/weather` from the browser? Describe the response and explain why.