Lab 9
AWS EC2 with Node

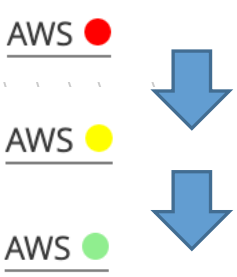
CSCI2720 Building Web Applications

Dr. Chuck-jee Chau
chuckjee@cse.cuhk.edu.hk

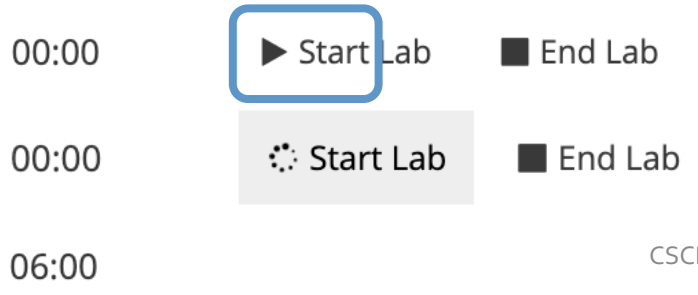
Agenda

- Setting up EC2 instance
- Allowing network access
- New user for SSH
- Basic Node server with Mongo
- Building React app and serve it in Node

Using AWS



- After registering with our AWS Academy invitation, you are entitled US$100 credits for trying selected AWS features
- Using ***Google Chrome***, log in at *https://awsacademy.instructure.com*
- Go to ***AWS Academy Learner Lab » Modules » Learner Lab***
  - You can read the student guide yourself
- Choose ***Start Lab***, and then the AWS dot will become green soon



Using AWS

- Every time you start a lab, you will be able to access the selected AWS features
  - There is usually a countdown clock of some hours before your session ends
  - You can start the lab infinite times, before your credits are used up
- Click on the ***AWS word*** to see the AWS console
  - You are free to use any of these services within this semester, even if not instructed in class
- Choose ***Compute » EC2***
  - EC2 — Elastic Compute Cloud

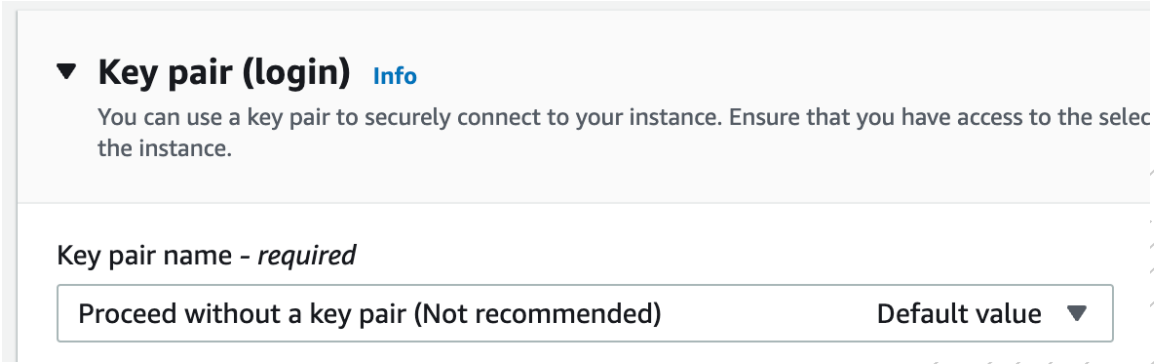
EC2 Instance

- You can consider EC2 instances as virtual machines in the cloud, which you can have full administrative power
- "Serverless architecture"
  - Everything has a cost: hardware configuration, computation time, bandwidth
  - High flexibility and scalability in resource management

EC2 Instance

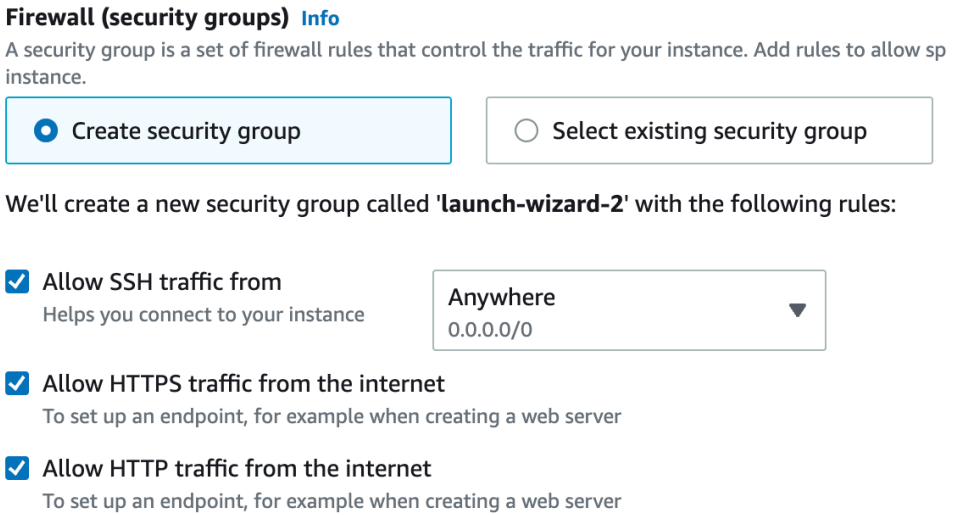
1. Click on the "Launch Instance" button
   - In the search box, type **csci2720**
   - Press Enter to search
   - There should be 1 result in Community AMIs

2. Scroll down, at **Key pair (login)** choose to *proceed without a key pair*
   - We will use SSH with password to connect



EC2 Instance

3. At Network settings, make sure these are all enabled: **Allow SSH from Anywhere**, **Allow HTTPS traffic**, **Allow HTTP traffic**



4. Then you can choose the big **Launch instance** button
5. Your instance will start soon
   - It will be stopped when you end the lab

EC2 Instance

6. To connect, click on the instance ID to show details and actions
   - The instance is only ready when showing 2/2 checks passed under **Status Check**
7. Choose **Connect**
8. Under **EC2 Instance Connect**, type **ubuntu** as the username, then **Connect**

- In case you run into AWS server errors, contact AWS Support for prompt help: *https://support.aws.amazon.com/#/contacts/aws-academy*

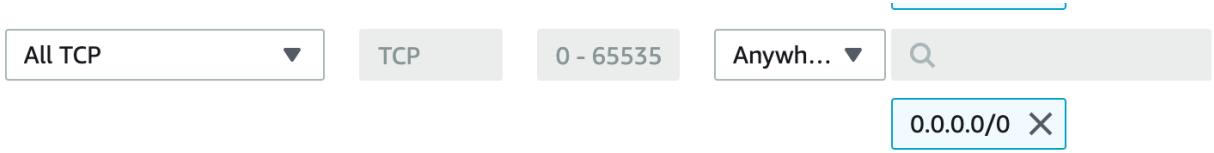
The CSCI2720 VM

- This server is running on Ubuntu 22.04 LTS
  - Pre-configured with Node v18 and MongoDB server 5.0
- You own the admin rights
  - **sudo** is available for the **ubuntu** user
- Feel free to configure the VM for your project, but *please don't break the settings*
  - Don't use up all US$100 before you finish the course project!
- If you need further help with Ubuntu Linux, try this:
  - https://www.google.com/search?q=ubuntu+crash+course

Allowing Network Access

- By default, the EC2 machines only accept inbound SSH traffics

1. Now return to the EC2 instance summary, choose the **Security** tab in the bottom, and click on the security group name
2. Choose **Edit Inbound Rules**
3. Add *All TCP* as type, *Anywhere-IPv4* as source, then save



- This makes sure all TCP ports are available from outside

New User for SSH

- You don't want to access this web interface every time to use the VM
- We will create new superuser with SSH access

1. Type the command
   `sudo adduser username`
   - This is a user you can use, so with a *username* you remember

```
ubuntu@ip-172-31-30-211:~$ sudo adduser chuckjee
Adding user `chuckjee' ...
Adding new group `chuckjee' (1001) ...
Adding new user `chuckjee' (1001) with group `chuckjee' ...
Creating home directory `/home/chuckjee' ...
Copying files from `/etc/skel' ...
New password:
```

2. Follow the screen instructions, and use a proper password
3. Add this user to the "sudo" group
   `sudo usermod -aG sudo username`
   - Note: 2nd "sudo" here is the group name, and *username* is the one you used before

- You can add user for all your groupmates later

Allow Password Login via SSH

- By default, SSH only allows logging in using public key
- Here we will accept password instead (less secure!)
- Use this command, edit the sshd_config file

  `sudo nano /etc/ssh/sshd_config`
- Press **Ctrl+W** to look for `PasswordAuthentication`, change the ending from **no** to **yes**
- Press **Ctrl+X** to save
- Restart sshd to take effect

  `sudo systemctl restart sshd`

```
ubuntu@ip-172-31-30-211:~$
```

i-06e3eeefd8eb3dd86

Public IPs: 52.72.246.235 Private IPs:

Your new username, and your VM's public IP

New User for SSH

- Using your favourite SSH client, connect to your public IP with the new username
  - You can use Terminal (macOS) or cmd (Windows)
  - `ssh` *`username@publicIP`*

```
chuckjee@mba2022 ~ % ssh chuckjee@3.91.23.18
The authenticity of host '3.91.23.18 (3.91.23.18)' can't be established.
ED25519 key fingerprint is SHA256:jqQJn8Zr2WHowX6bL0XcBX8/CLOgZnaMWeU83RSW1A8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

  - Other clients include *PuTTy*, *MobaXterm*, *Termius*, etc.
- Prefix **`sudo`** for any commands requiring admin right
- Try **`sudo -i`**⏎ to stay in the superuser environment
  - **`exit`** to end

Node and Mongo

- Set up a basic Node app with Mongo to test
  - `mkdir lab9; cd lab9`
  - `npm init`
  - `npm install express mongoose`
  - `nano server.js`

```
const express = require('express');
const app = express();
const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/new');
const db = mongoose.connection;
db.once('open', function() {
console.log("Connection is open...");
});

app.get('/*', (req,res) => res.send("Success"));

const server = app.listen(80);
```

- Start the Node server by
  - `sudo node server.js`
- Try and visit your public IP address in browser!

Getting React into Express

- Kill the Node server with **Ctrl+C**
- Return to your home directory

```
cd
```

- Set up a new React app

```
npx create-react-app my-app
```

- After setting up, try the default React app

```
cd my-app
npm start
```

- In your browser, visit your public address but add **:3000** at the end for the specific port of the React dev server

Getting React into Express

- Edit the source code App.js

```
nano src/App.js
```

- Find the line between ** ** and ** **, replace it with **CSCI2720**
- Save the file and try **`npm start`** again
  - Note: **`npm start`** only gives you a development server
- Add this line in **`package.json`**, after the line of **`"name"`**

```
"homepage": ".",
```

- Now make a production build

```
npm run build
```

Getting React into Express

- Return to the original Express app

```
cd ../lab9
```

- Set up a symbolic link to the React app build

```
ln -s ../my-app/build app
```

- In the Express JS file, change the original route to something like this:
  - `app.get('/api/*', …)`
  - This path **`/api`** will be matched other than anything React (router) would match
- Add this after the Express routes

```
app.use(express.static('app'));
```

- Now, start Node and try!
  - `http://(publicIP)`
  - `http://(publicIP)/api/trying`

That's it!

- Hope that everything works fine!
  - In AWS Academy, choose "Stop lab" when you are done
  - All EC2 instances will be stopped
- You can use this VM for your project work
  - Next time you only need to choose the instance and start it, no more setup required
- Remember: don't use up the credits before you finish the course project!
- The AWS Academy access is available until ***end of December***, so you can use remaining credits for your own exploration

IMPORTANT

- You may share your AWS VM account with groupmates
- Yet, NEVER keep your individual course assignment files in any shared area!