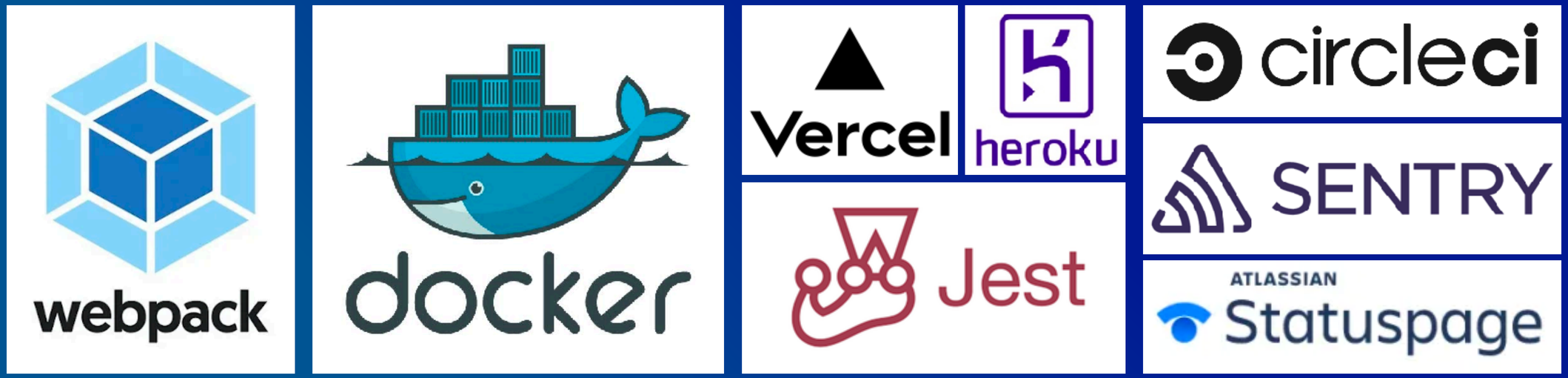
12. Things You Need to Know about Production



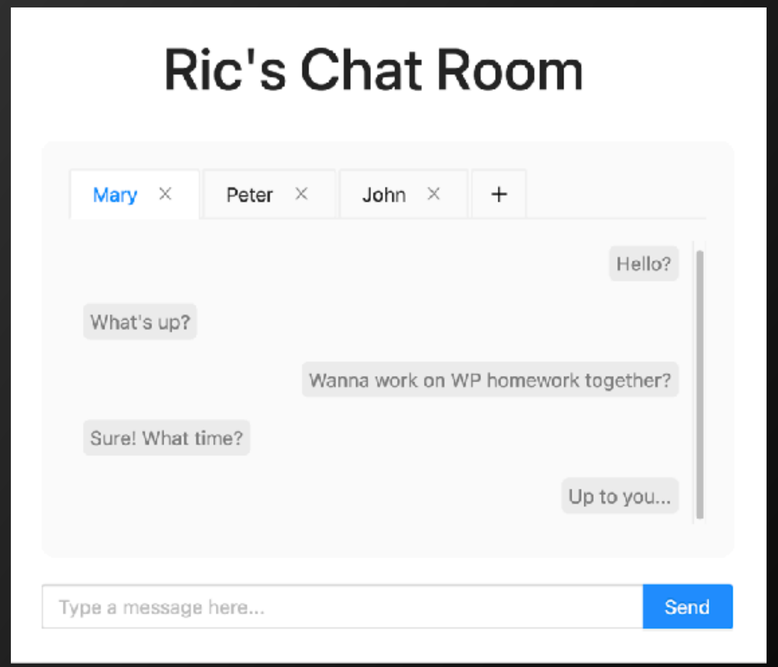
Ric Huang / NTUEE

(EE 3035) Web Programming

Project 開發完成後，
下一件重要任務，
就是要 deploy 給大家使用。

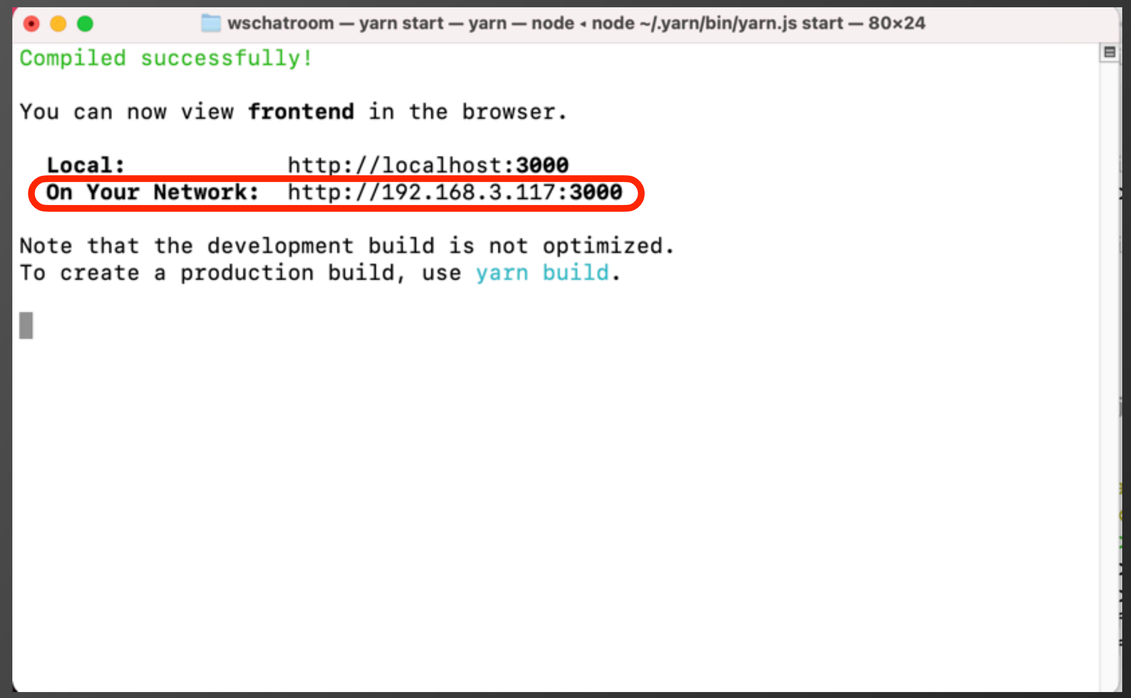
先拿我們做過的全端 App
如 HW#7/8 來做例子

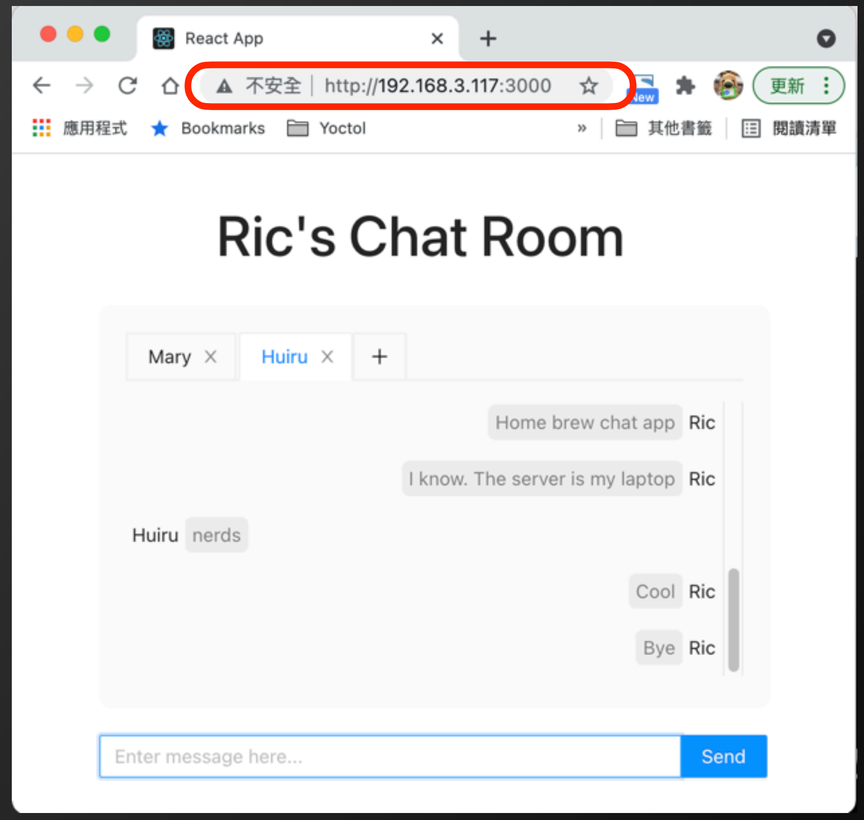
你可以跟你的同學在兩台機器同時
使用這個 App, 然後對話嗎？



一個土炮的 deployment 實驗

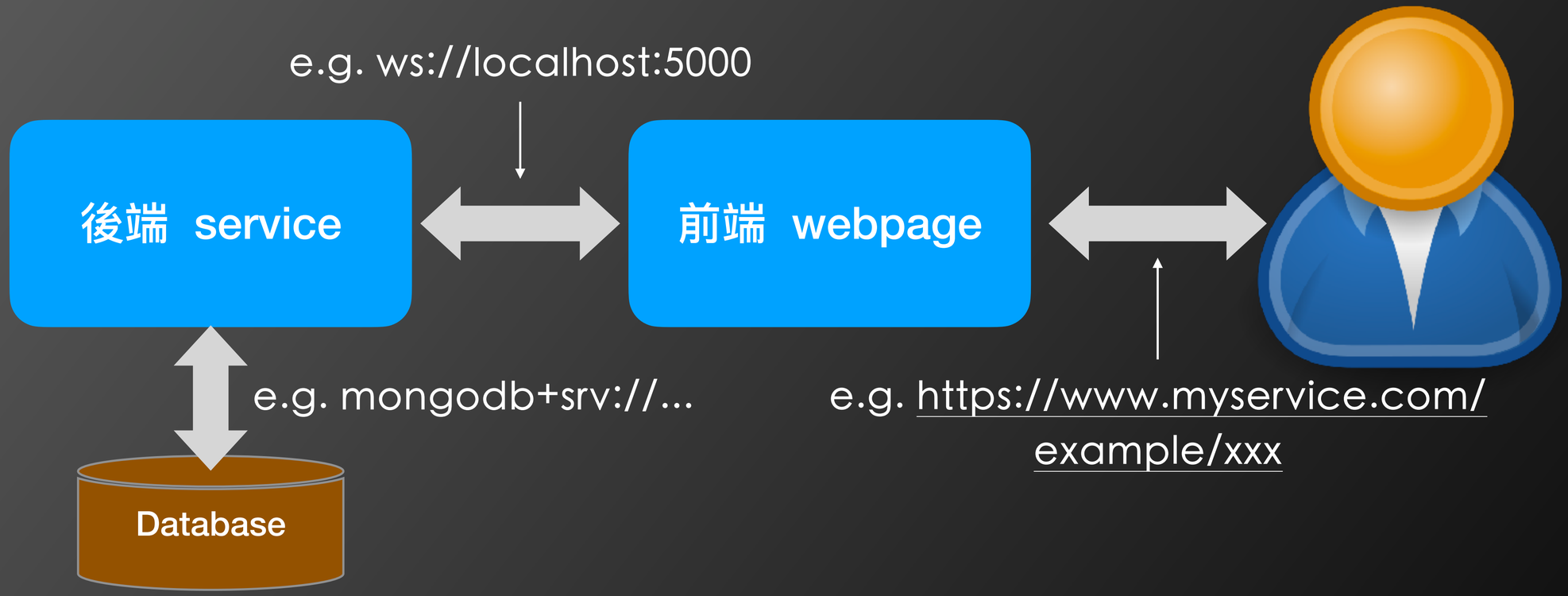
先把你的 HW#7/8 跑起來
注意前端 yarn start 後的訊息





在同個網域下，打開
browser 輸入該網址試試看！

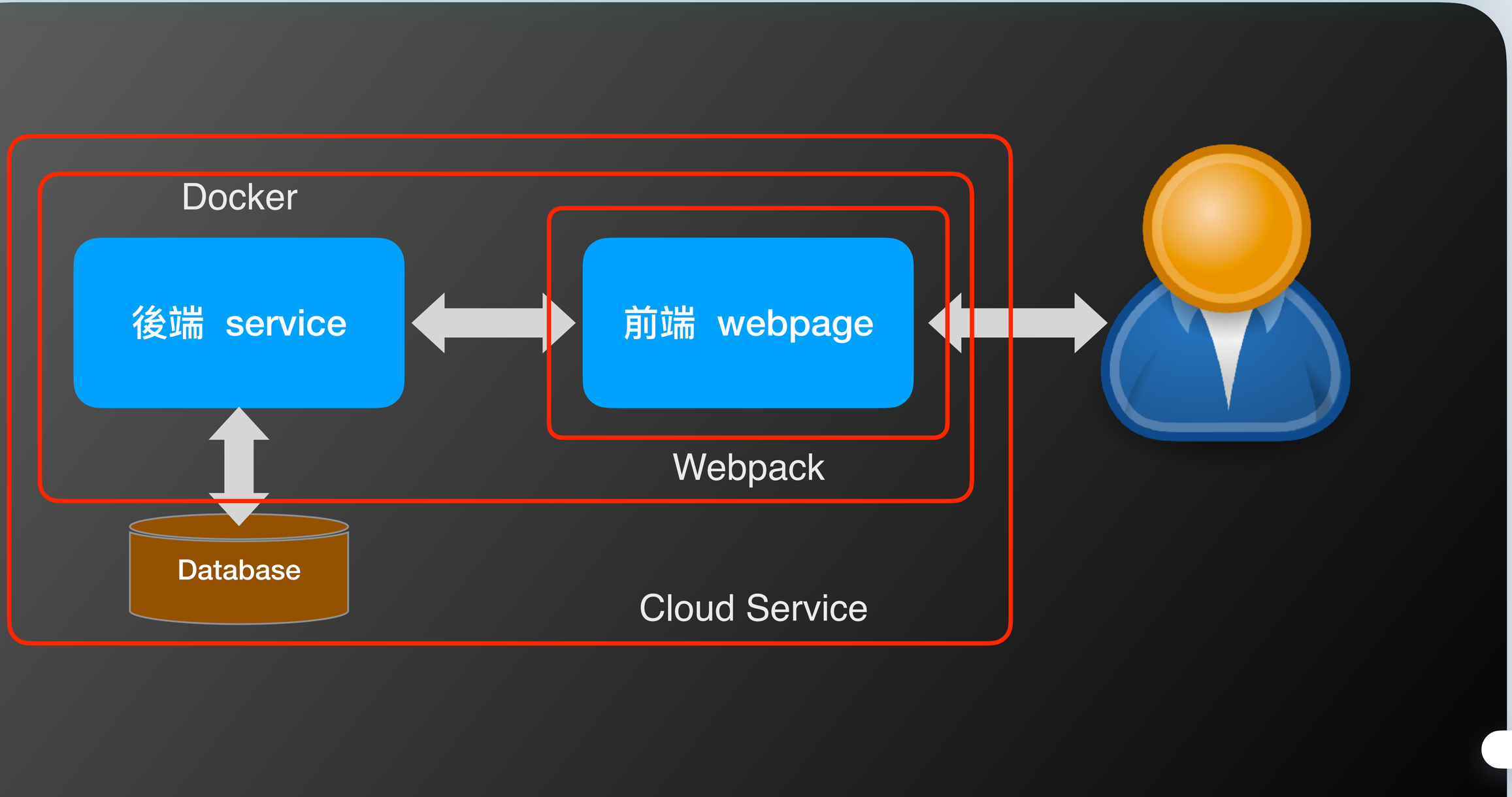
一個簡單的 Deployment 示意圖



所以 deployment 的問題簡單的說，

就是你想把你前後端的程式跑在哪裡？DB 要放哪裡？

Looking Forward...



曾幾何時，網頁已經不再是單純的網頁...

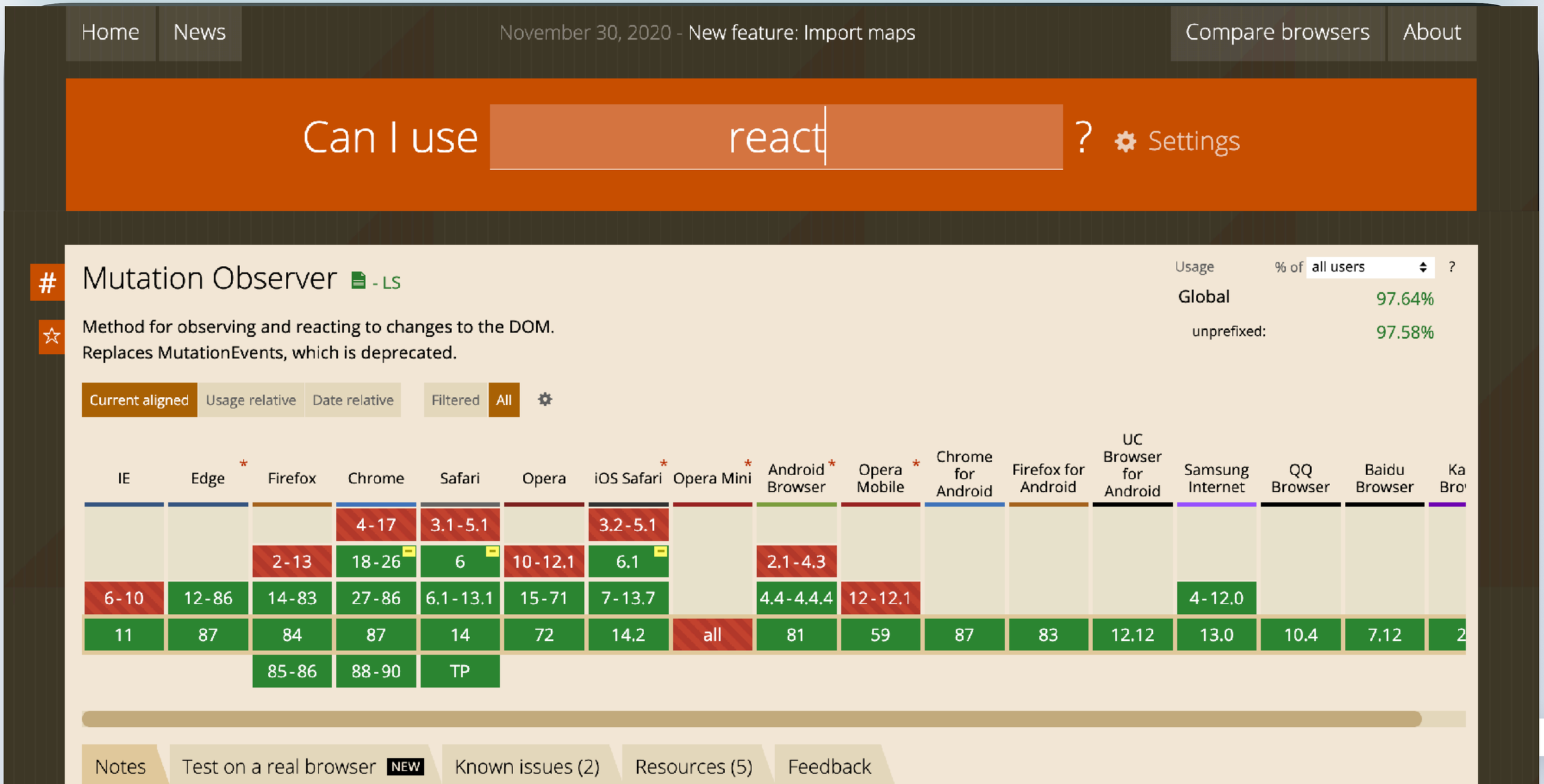
- 從前從前，我們認為網頁就是 HTML，瀏覽器就是讀懂 HTML 就好
- 後來，JavaScript 出現，Java & Python 也想試著插一腳，當然還有 CSS, php, Ruby, .net, C#... 等
- 更別提近年來 JS, CSS, 甚至 HTML 都不斷的推陳出新
- 還有各種前後端的框架，族繁不及備載

你使用的瀏覽器可以打開這個網頁嗎？

還有這個令人上火的提醒訊息..

還好，MS 在今年就會讓 IE 走入歷史...

Can I use _____?


Home
News
November 30, 2020 - New feature: Import maps
Compare browsers
About
Can I use react ? Settings
Mutation Observer - LS
Method for observing and reacting to changes to the DOM.
Replaces MutationEvents, which is deprecated.
Usage
% of all users
Global 97.64%
unprefixed: 97.58%
Current aligned
Usage relative
Date relative
Filtered
All
IE
Edge
Firefox
Chrome
Safari
Opera
iOS Safari
Opera Mini
Android Browser
Opera Mobile
Chrome for Android
Firefox for Android
UC Browser for Android
Samsung Internet
QQ Browser
Baidu Browser
4-17
3.1-5.1
3.2-5.1
2-13
18-26
6
10-12.1
6.1
2.1-4.3
6-10
12-86
14-83
27-86
6.1-13.1
15-71
7-13.7
4.4-4.4.4
12-12.1
4-12.0
11
87
84
87
14
72
14.2
all
81
59
87
83
12.12
13.0
10.4
7.12
85-86
88-90
TP
Notes
Test on a real browser NEW
Known issues (2)
Resources (5)
Feedback

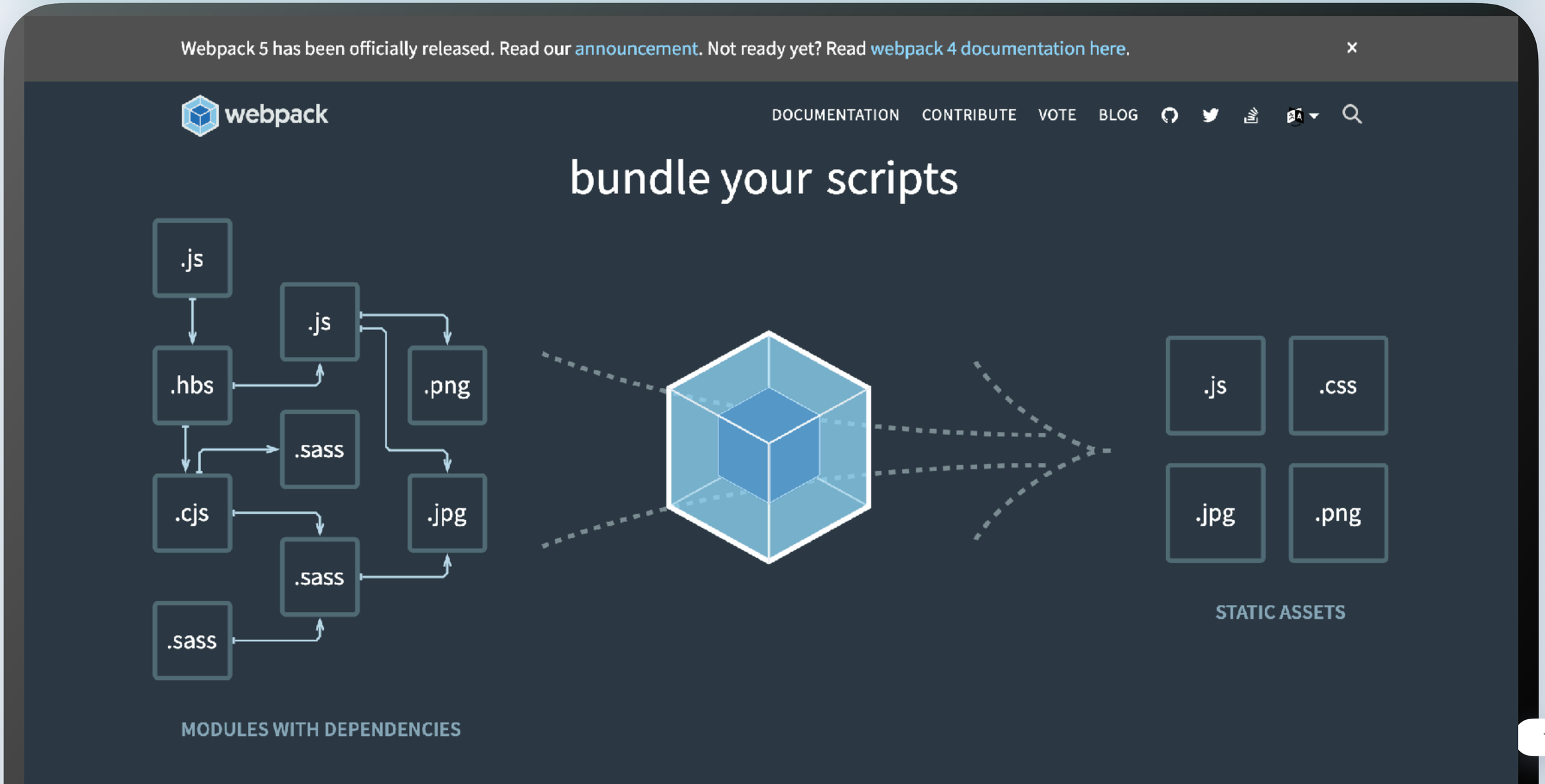
Recall: Babel — The Modern JavaScript Transcompiler

- (From Wiki) Babel is a free and open-source JavaScript transcompiler that is mainly used to convert ECMAScript 2015+ (ES6+) code into a backwards compatible version of JavaScript that can be run by older JavaScript engines.
- It was originally created by Sebastian McKenzie at age of 17 (2015). He then joint Facebook at 18. He was also the author of yarn.
- Read this for more of his story.

Babel 讓你有個舒服的開發環境，
可以使用各種版本的語言與套件，
不用太擔心衝突的問題...

但問題是，當你要發佈你的服務的時候，
如何確保使用者的 browser 有安裝適當
的套件，你的(前端)程式碼不會因為這些
preprocessors 而變成過於肥大，
造成瀏覽時候的負擔？

What is “Webpack”?



What is “Webpack”?

- 要使用 webpack，就是在 package.json 的 scripts 加上：

```
"scripts": {
   "build": "webpack --config webpack.config.js”
}
```

- 然後在發布之前，跑 “yarn build”，透過 webpack 程式以及 webpack.configuration.js 這個 script, 把 (前端) src 底下程式的 dependency tree 建立起來、進行編譯、打包、甚至 uglify, compress, 放到 dist 這個目錄，準備好可以發布

Webpack Configuration(ref)

雖然 Webpack v4.0 以後 (now v5.0) configuration file 已非必須，但你可以透過底下 properties 來設定 webpack:

- Entry: 建立 dependency tree 的起始點，default 是 "src/index.js"
- Output: 輸出的 "bundles" 存放目錄以及檔案名稱，default 是 "dist/main.js"
- Loaders: 用來指定打包除了 JS & JSON 以外的檔案的方式
- Plugins: 執行 bundle 最佳化、asset 管理、注入環境變數
- Mode: "development", "production" or "none"
- Browser Compatibility: 呃... for those old IE browsers...

Webpack Configuration File (Sample)

```
const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {
  entry: './path/to/my/entry/file.js'
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'my-first-webpack.bundle.js'
  }
  module: {
    rules: [
     {
       test: /\.js$/,
       use: {
         loader: 'babel-loader',
         options: {
           presets: ['@babel/preset-env']
         }
       }
     }
    ]
  },
```

Webpack Configuration File (Sample)

```
  plugins: [
    new CleanWebpackPlugin(),
    new HtmlWebpackPlugin({
      template: './views/index.html',
      filename: './index.html',
      minify: {
       collapseWhitespace: true,
        removeAttributeQuotes: true,
        removeComments: true,
        ...
    }})
  ],
  optimization: {
    minimizer: [
      new TerserPlugin({
        terserOptions: {
          ...
        },
        extractComments: false
  })]}
};
```

Webpack。Conclusion

- Rollup.js: 近年也很多人使用的另外一套 webpack 工具
- 雖然 webpack 解決了專案在發佈時跨平台、module dependencies 等問題，但設定 webpack 還是一件十分繁瑣的事，並且在發佈時還要考慮架設 server 等問題
- 事實上，除非你想要自己管理機器、建置 Linux 伺服器，否則如果使用雲端的服務，很多服務都已經把 deployment 內建在他們的服務選項裡面，你可以不用自己去設定 webpack 了

Deployment

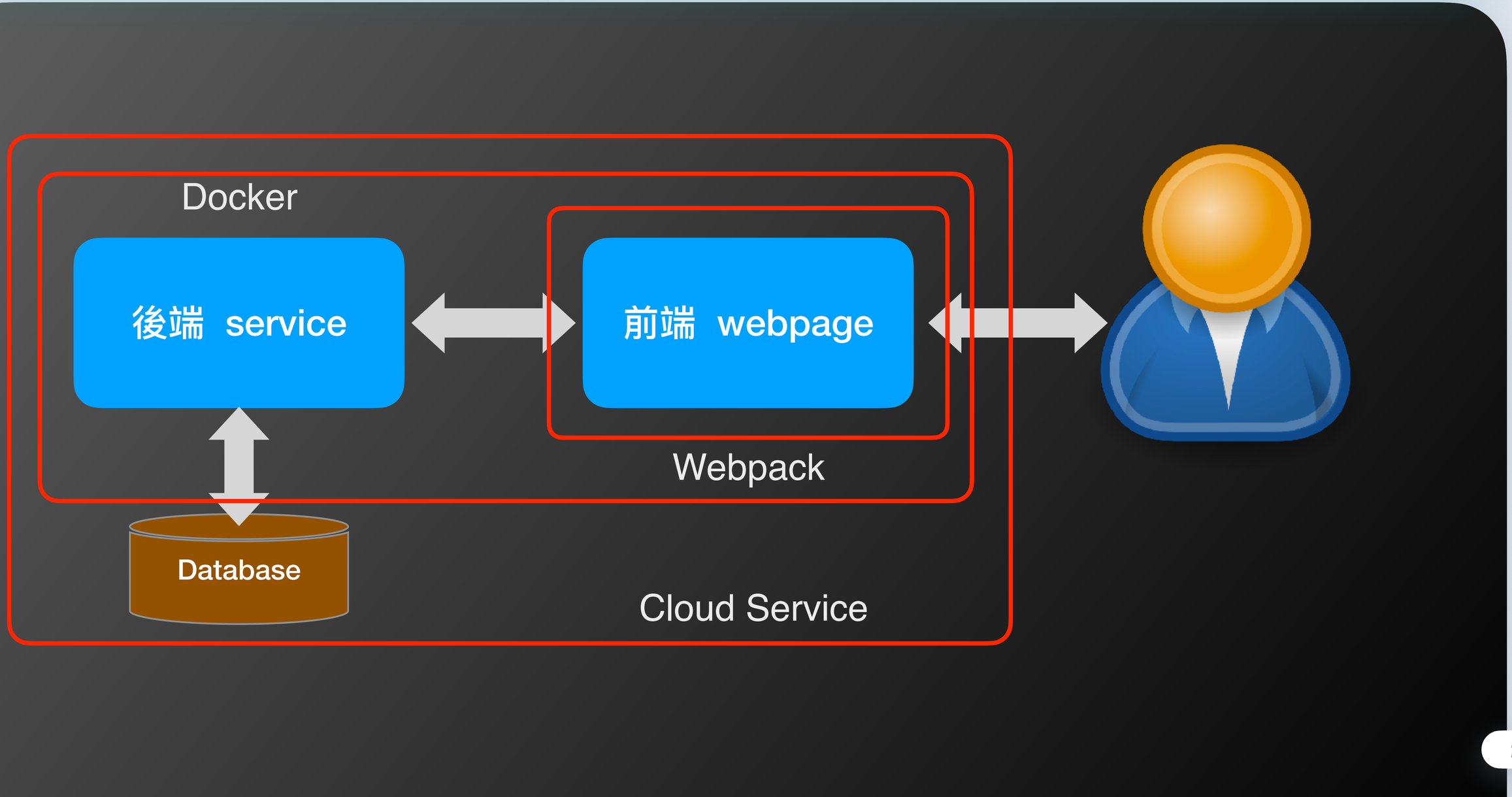
- "Deployment" 顧名思義就是把你的 web service programs "deploy" 到一個 cloud/web server, 然後讓使用者可以透過 web browser + URL 去使用你的服務
- Deployment 的做法有很多種：
  - 大型的 IaaS providers: AWS, MS Azure, Google Cloud, IBM Cloud... etc. => 十分完整強大，best for production use (當然也比較貴), 用起來也複雜很多
  - 強調簡單/直覺的 cloud services: Heroku, DigialOcean, Vercel, Railway, (GitHub)... etc. => 多為 startups (unicorns), 真的簡單/便宜很多
  - 自建 server: 找個固定 IP, 申請個 domain name, 把 web programs deploy 到那邊

Deployment Tutorials

- 大部分的雲端平台都有提供一定的免費額度。
  👉 See TA’s tutorials on Deployment
  - By 110-1 助教：Heroku, GCP, GitHub
  - By 111-1 助教：Railway

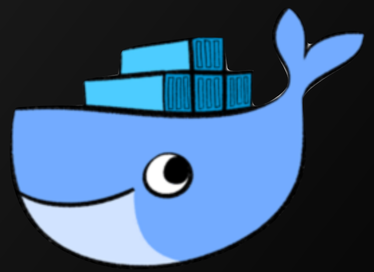
Note: 與 HW#9 & Final Project 相關，一定要會！

Recap...

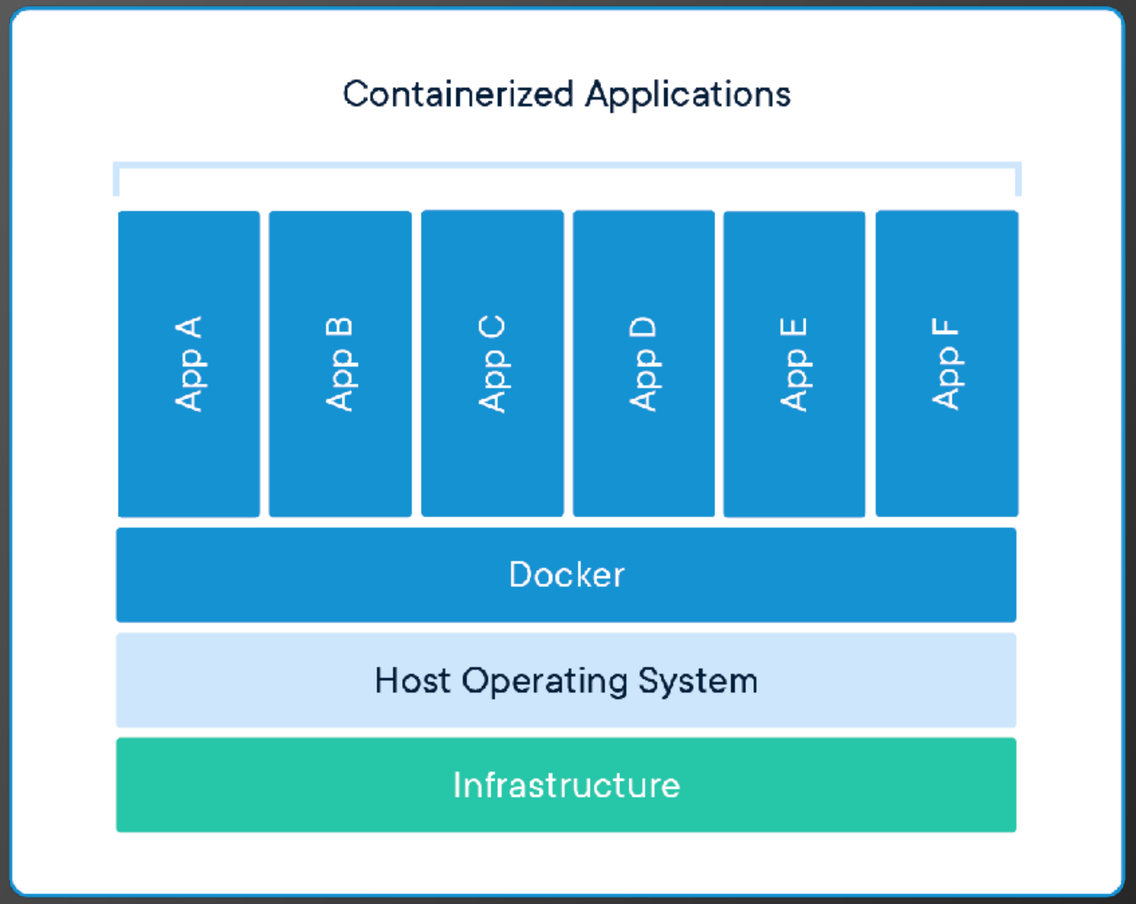

Docker
後端 service
前端 webpage
Webpack
Database
Cloud Service

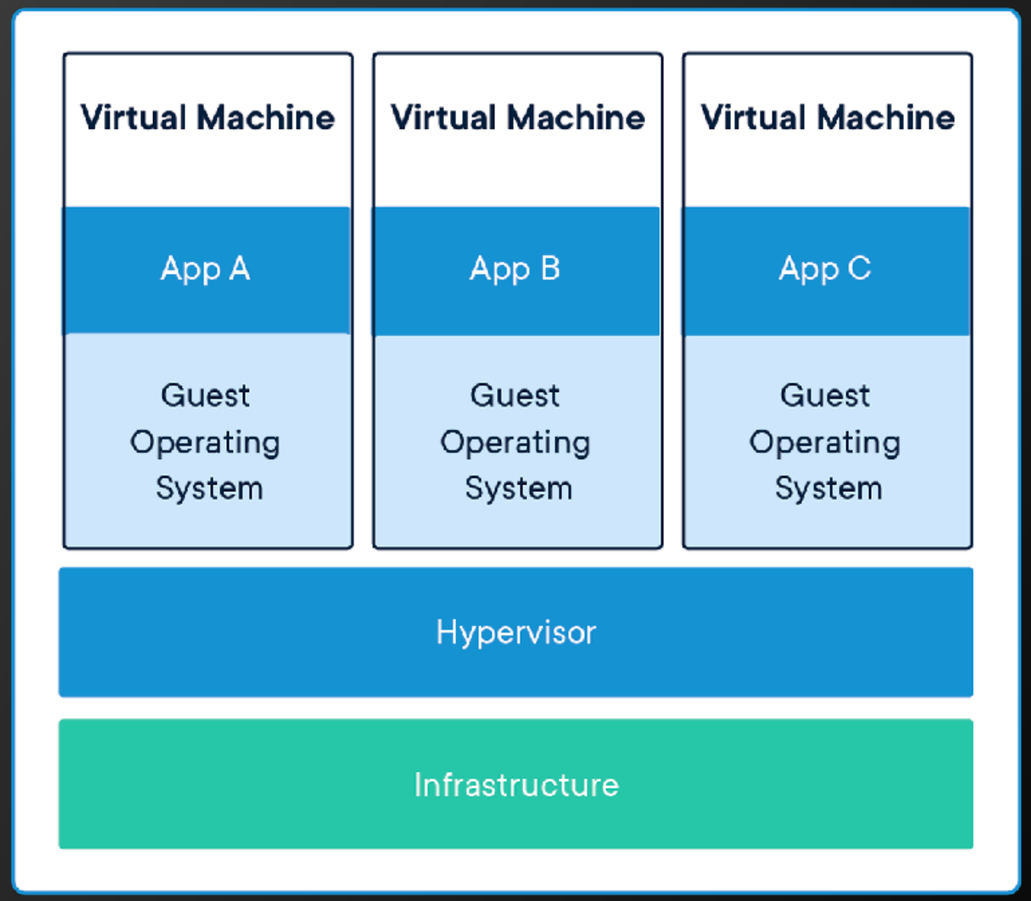
電腦資訊的進步，總是朝向簡化的方向前進

- 最遠古之前，我們只能使用貴森森的 Sun 工作站 (early 90s)
- 很久很久以前，我們必須要用好幾片磁碟片才能安裝 Linux (late 90s)
- 後來出現了很多幫你包裝好的 Linux distributors (early 00s)
- Virtual Machine 的流行，讓我們使用 Linux 不用重開機 (late 00s)
- 雲端服務的成熟，讓我們甚至不用買自己的伺服器 (early 10s)
- 最後，container 技術的出現，讓我們可以不用擔心環境不統一的問題 (late 10s)



Container vs. Virtual Machine





Learning Docker

- 先確定你有安裝好 docker
- Docker docs 裡頭其實有一個還不錯的 tutorial, 助教接下來會帶大家 go through it

最後，如果你想要建造一個比較專業的 project 至少底下幾件事情你要去做：

1. 學習並使用 authentication
2. Follow a good flow (e.g. Git Flow)
3. Coding style & linter
4. (先)寫測試
5. CI/CD 環境
6. 自動錯誤偵測
7. 服務使用狀態監測
8. 使用者行為追蹤

1. 學習並使用 authentication

bcrypt 是 node.js 生態系裡頭一個輕量，但很受歡迎的密碼加密套件，每週有近 60 萬次的下載量

A very simple example

- Again, let's make a simple backend from scratch

```
> mkdir simple_passwd && cd simple_passwd
> yarn init -y
> Add the following to "package.json"
  "scripts": {
    "start": "nodemon --exec babel-node src/index.js"
  },
> Create a new file ".babelrc"
  {
    "presets": [
      "@babel/preset-env"
    ]
  }
> yarn add -D @babel/node @babel/core @babel/preset-env
> Create a new file "src/index.js"
  console.log("Hello, World!")
> yarn start  // 是否成功看到 Hello, World!
```

Adding bcrypt

- yarn add bcrypt
- Modify "src/index.js" to —

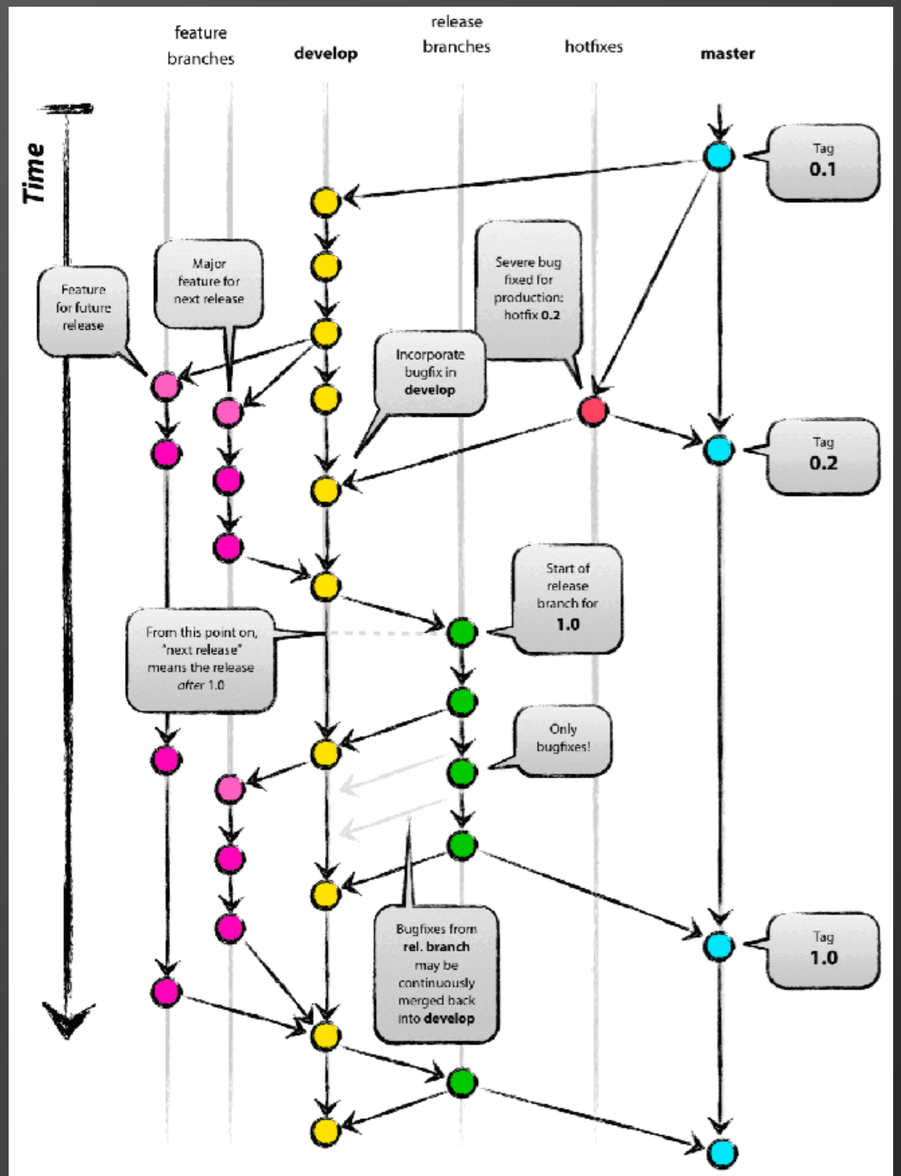
再跑一次看看！
hash 還是一樣嗎？

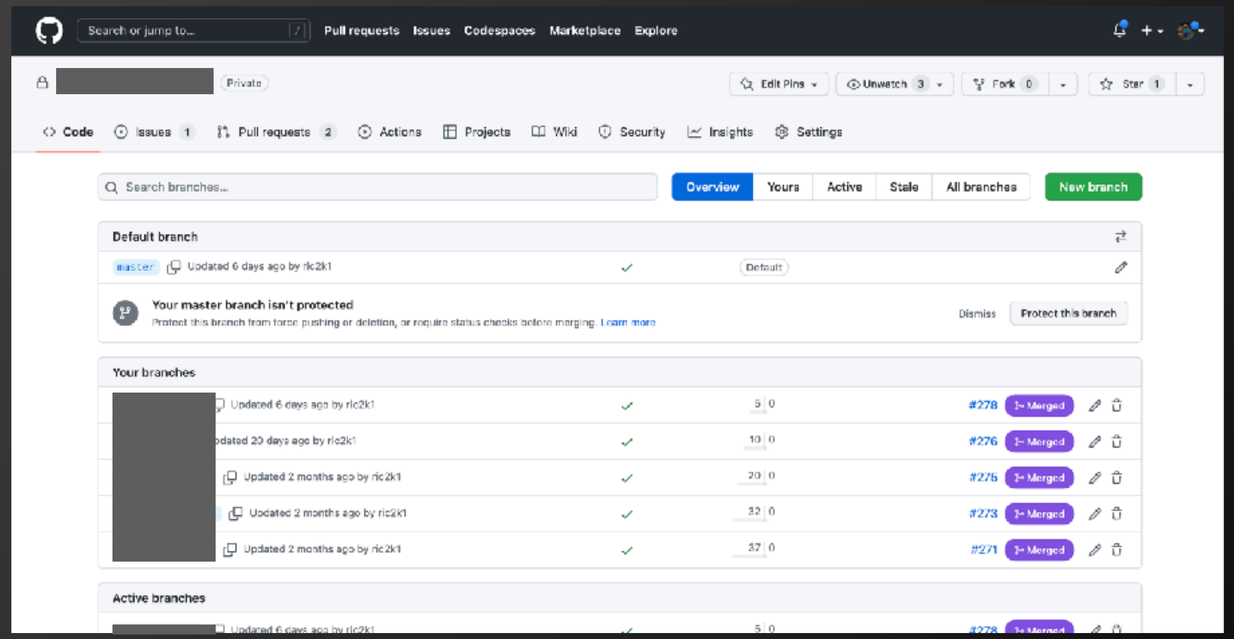
```
import bcrypt from "bcrypt";
const saltRounds = 10;
const myPassword = 'password1';
const testPassword = 'password2';
const myHash
='$2b$10$Ii0DAbr2qUVFfMElKVkaSec.fLTgaJITTCnFMn84G1uj4o.dk3WdC';
// for testing purpose
(async () => {
    const hash = await bcrypt.hash(myPassword, saltRounds)
    console.log(hash)
    console.log(myPassword)

    const res = await bcrypt.compare(myPassword, myHash)
    console.log(res)

    const res2 = await bcrypt.compare(testPassword, myHash)
    console.log(res2)
})()
```

Making it practical

- Of course, in real-life example, 我們不會把 hash key 放在程式碼裡
  - 把 hash key 以及 encrypted password 存放在 DB, 當 user 從前端輸入帳密之後，傳送到後端來比對，成功才可以登入，由後端送出後續的服務。

2. Follow a good flow (e.g. Git Flow)





- Important git commends:
  - branch, checkout, status, diff, merge, rebase,... etc.

3. Coding Styles and Linter

- 寫程式其實是很 ego 的事情
- 常常會對別人的 code 看不順眼
- 因此，規範一些 coding styles (e.g. 命名原則、大小寫、空白鍵、括號... etc)，可以有助於團隊合作的和諧
- 為了實踐這樣的規範，可以使用一些 linting tools or plug-ins 來進行檢查
- 一些有名的工具像是 ESLint, 以及大家廣為稱頌的 coding style 像是 AirBnB

4. 寫測試。先寫測試再寫 code

- TDD: Test-Driven Development
- Some popular testing tools for Javascript

Running Jest (ref)

- In package.json

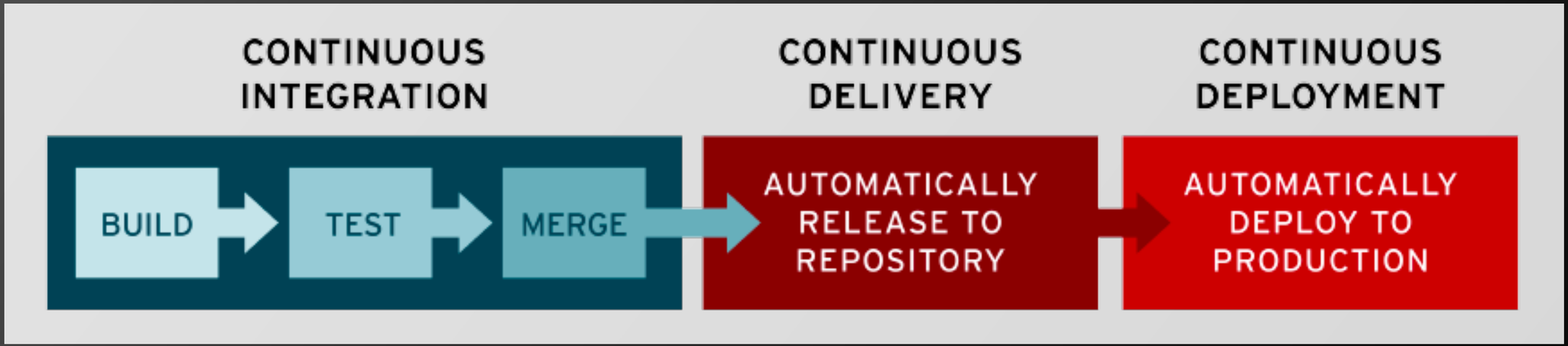
```
scripts: {
  "test": "jest"
}
```

- Write test file (.test.js)

```
test('Check the result of 5 + ', () => {
    expect(5 + 2).toBe(7)
})
```

5. Concept of CI/CD

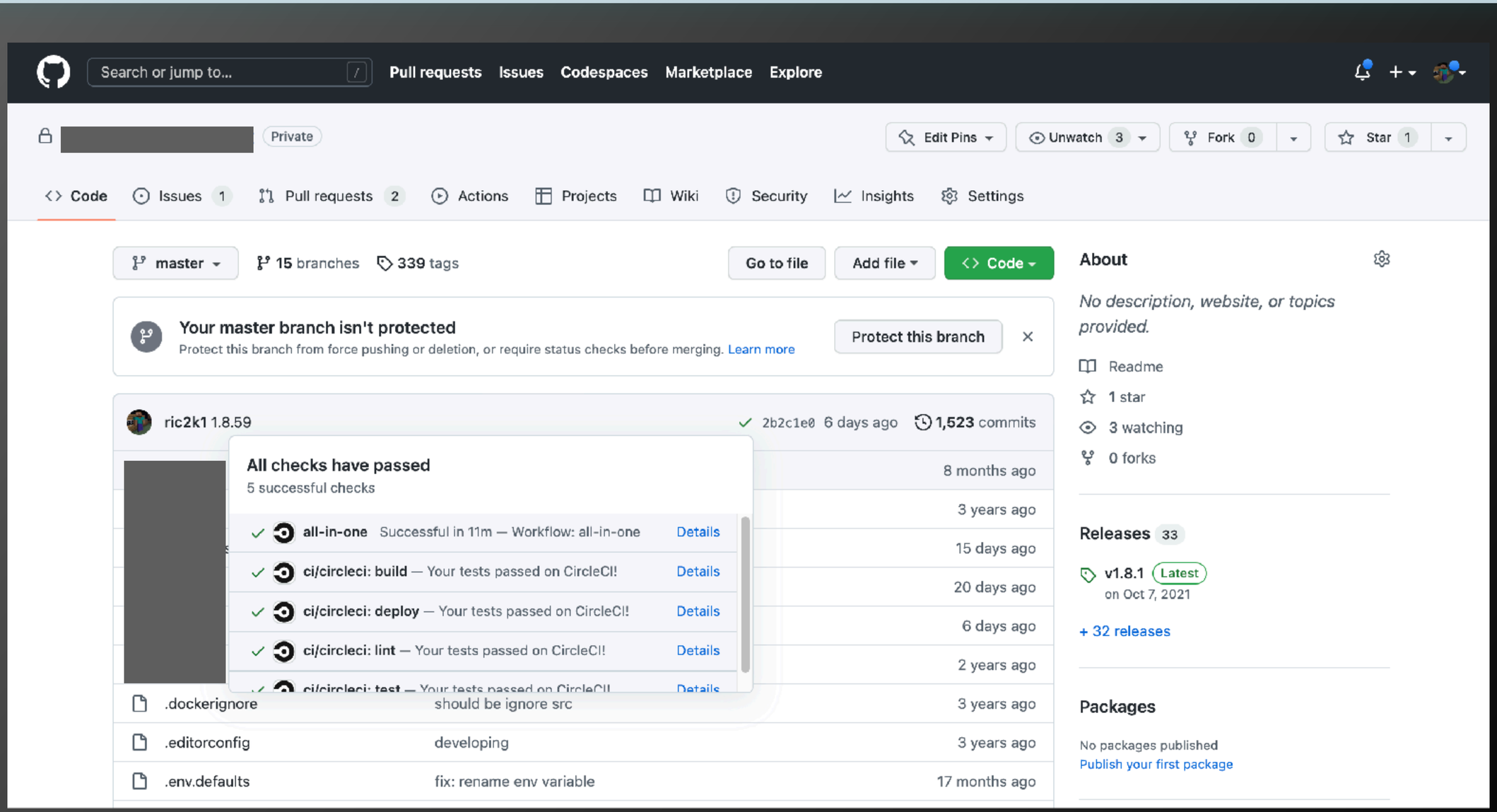
- CI/CD introduces ongoing automation and continuous monitoring throughout the lifecycle of apps, from integration and testing phases to delivery and deployment. Taken together, these connected practices are often referred to as a "CI/CD pipeline" and are supported by development and operations teams working together in an agile way with either a DevOps or Site reliability engineering (SRE) approach. (Ref)



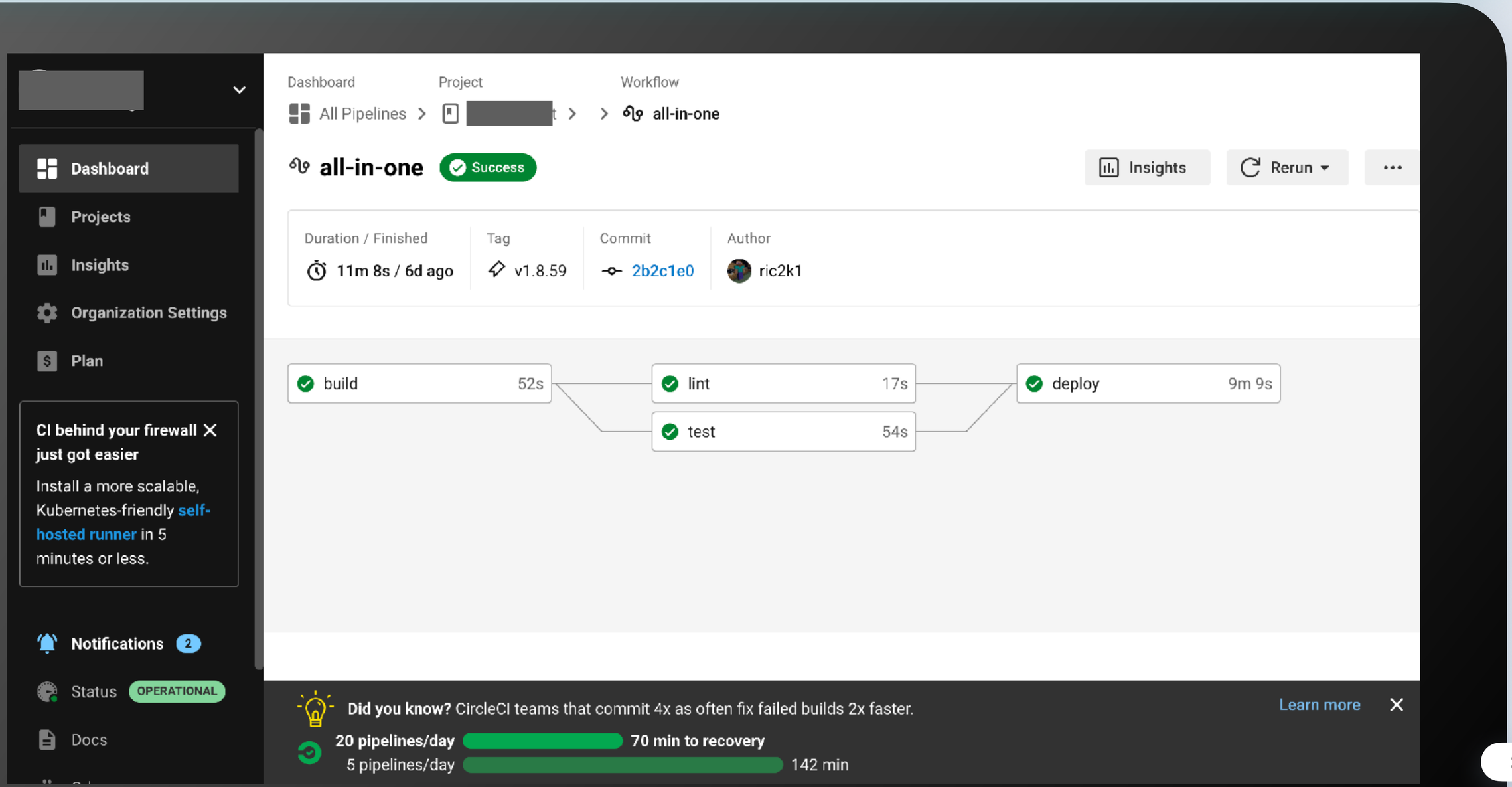
CircleCI for Continuous Integration and Delivery

- Online tutorial — https://circleci.com/docs/2.0/tutorials/
- Integrating CircleCI with GitHub, Slack…

Highly Integrated with GitHub



Highly Integrated with GitHub

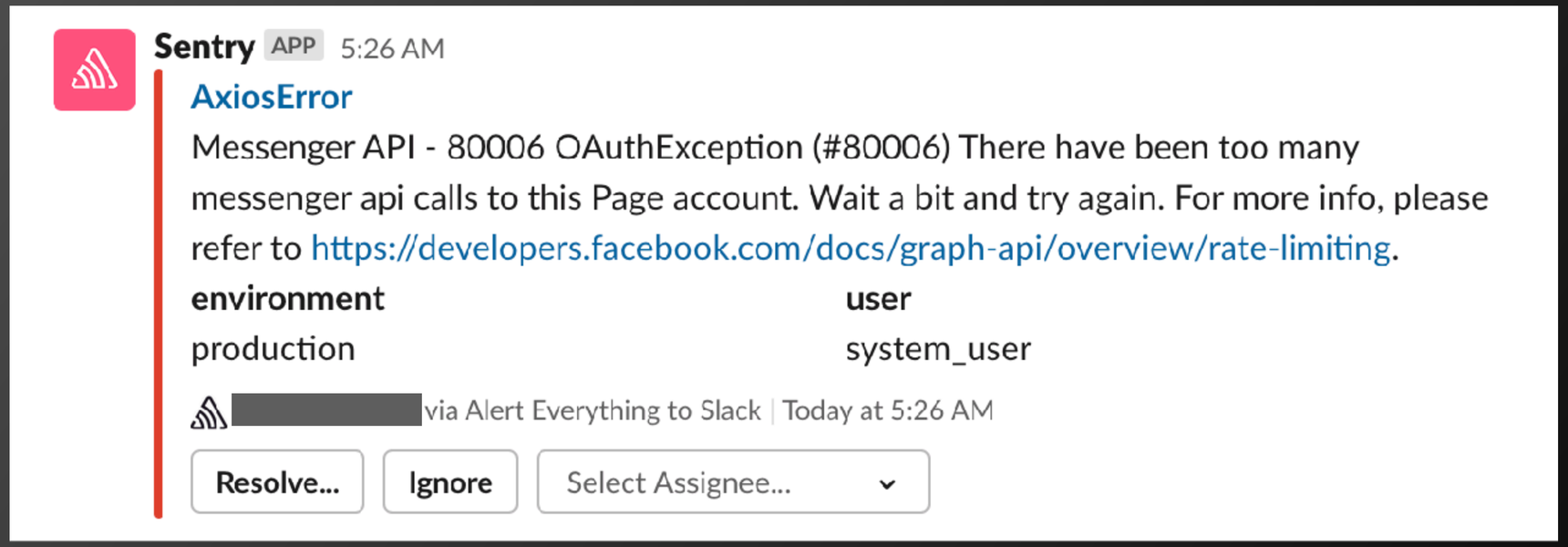


6. 自動錯誤偵測

當服務上線之後，要隨時注意
的是有沒有任何因為程式錯誤、
網路狀態、第三方服務中斷等
意外情況發生，導致服務
出問題，需要立刻處理

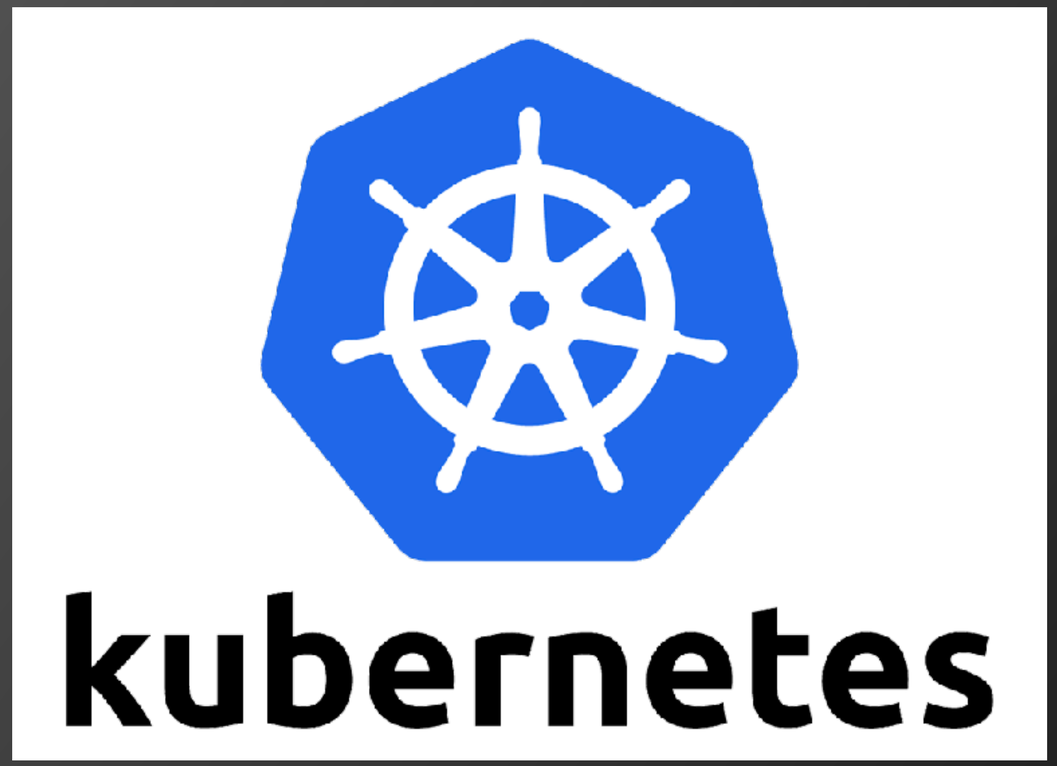
Sentry — Application Monitoring and Error Tracking

- Founded in San Francisco, 2012.
- Total funding: US$127M (was $66.5M in 2020)
- Sentry is an open-source platform for workflow productivity, aggregating data from errors and crashes across the stack in real time.
- Integrated with Slack:

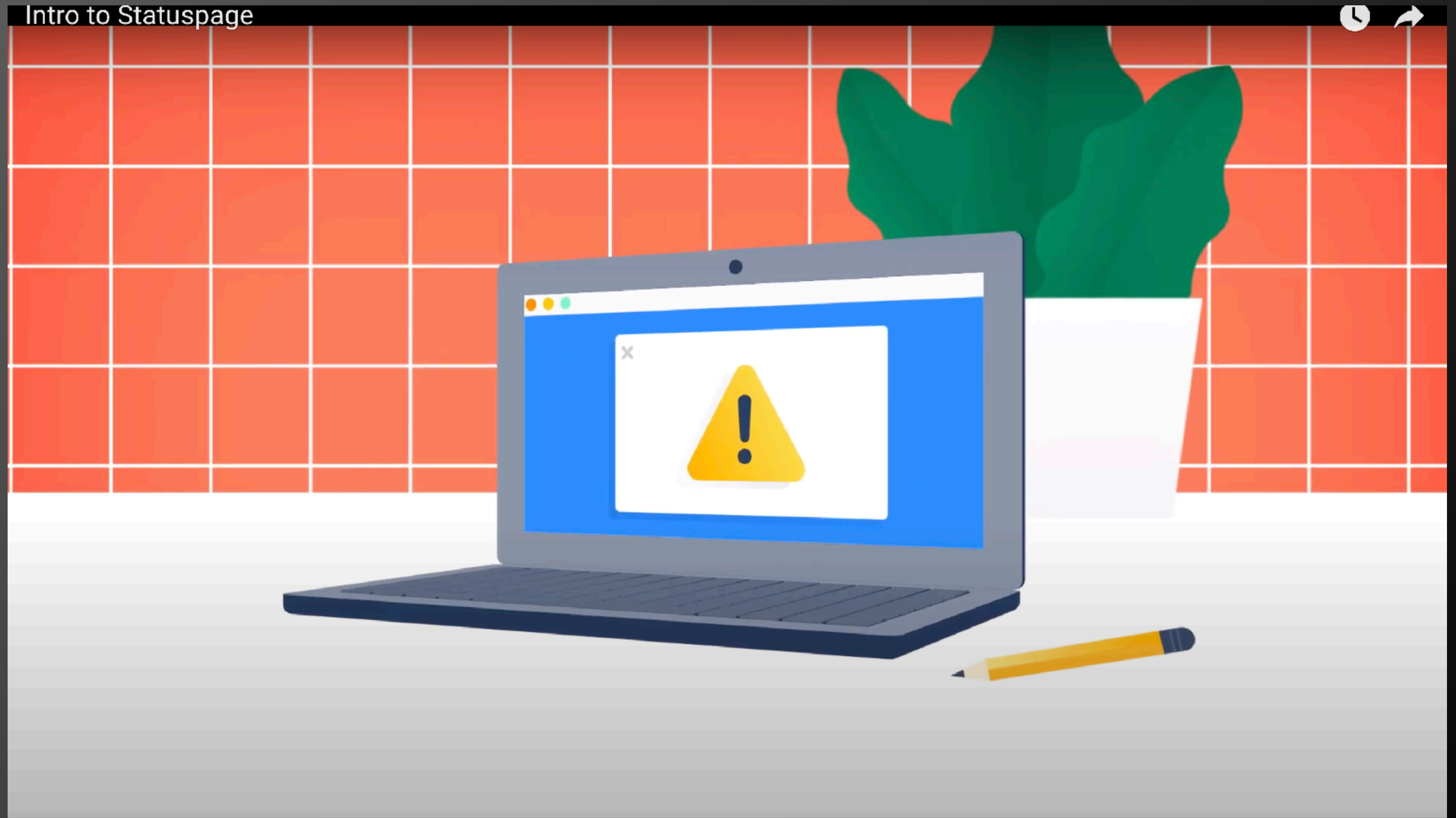


7. 服務使用狀態監測

一些基礎 pod/servers 管理的觀念與工具



Statuspage — Communicate Downtime with Customers



8. 使用者行為追蹤

最後，別忘了串接一些網頁追蹤服務，洞察使用者的行為，以作為未來改進服務的參考

That's it.

希望上完這學期的課後

你對於如何建造網路服務

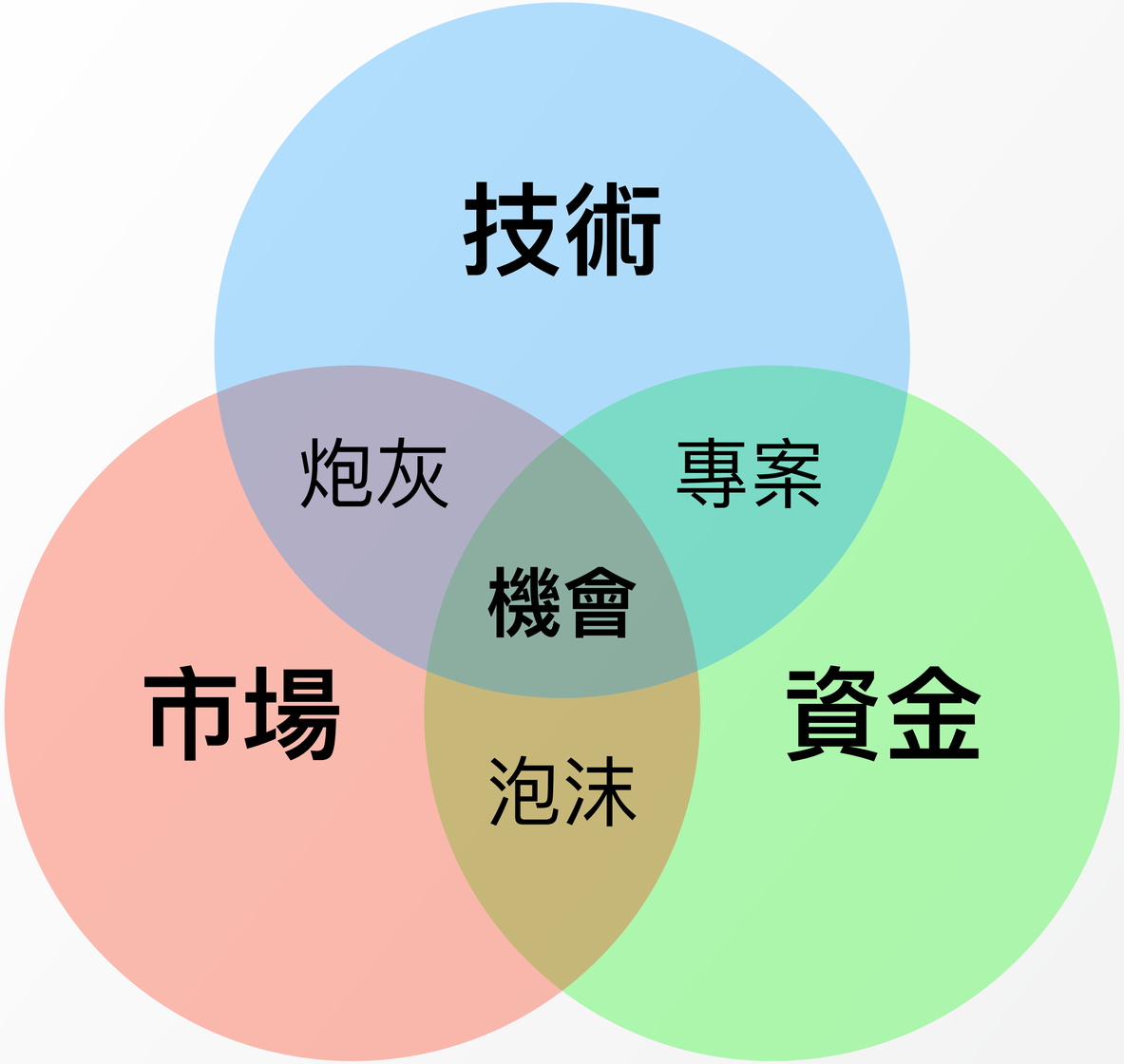
有基本上的了解

台灣軟體產業很弱
不是一天兩天的事
也不是一件容易解決的事

台灣軟體環境的一些討論...

- 台灣軟體產業為什麼弱？
  - (有此一說) 台灣年輕人缺乏 project-oriented / critical thinking 的訓練、只會解題 / 不會出題、社會歷練不夠 / 對於有價值的問題缺乏想像…
  - Yes or No
  - 「貧窮限制了你的想像！」 (well...)
  - 「有錢就是任性！」

創業三要素



- 你覺得現在(軟體)創投想要投資什麼？你熟悉嗎？
- 何謂市場？何謂市場人才？Market or Marketing?
- 為什麼要募幾千萬美金的錢？
- 政府可以做什麼？
- 學校可以做什麼？
- 你可以做什麼？

跟大家分享我的三個座右銘

1. 「理性的思考、感性的做決定。」
2. 「從最重要的事情開始做決定，
   不要怕犯錯。」
3. 「做人不要後悔。」

希望這門課幫大家開了一扇窗之後

大家在未來都有能力繼續自學

台灣軟體產業的未來就靠你們了！

感謝聆聽！

Ric Huang / NTUEE

(EE 3035) Web Programming