



# Kubernetes

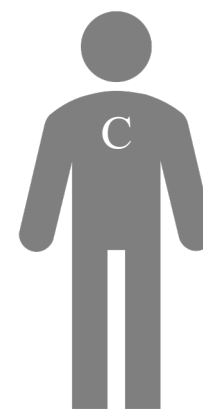
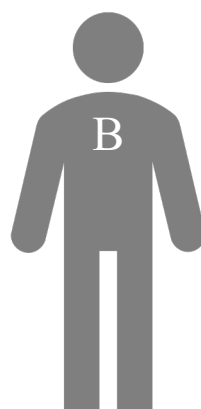
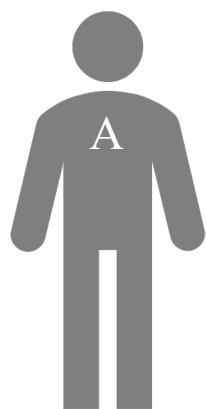
Samuel Luo 羅恩力

我想先說一個故事

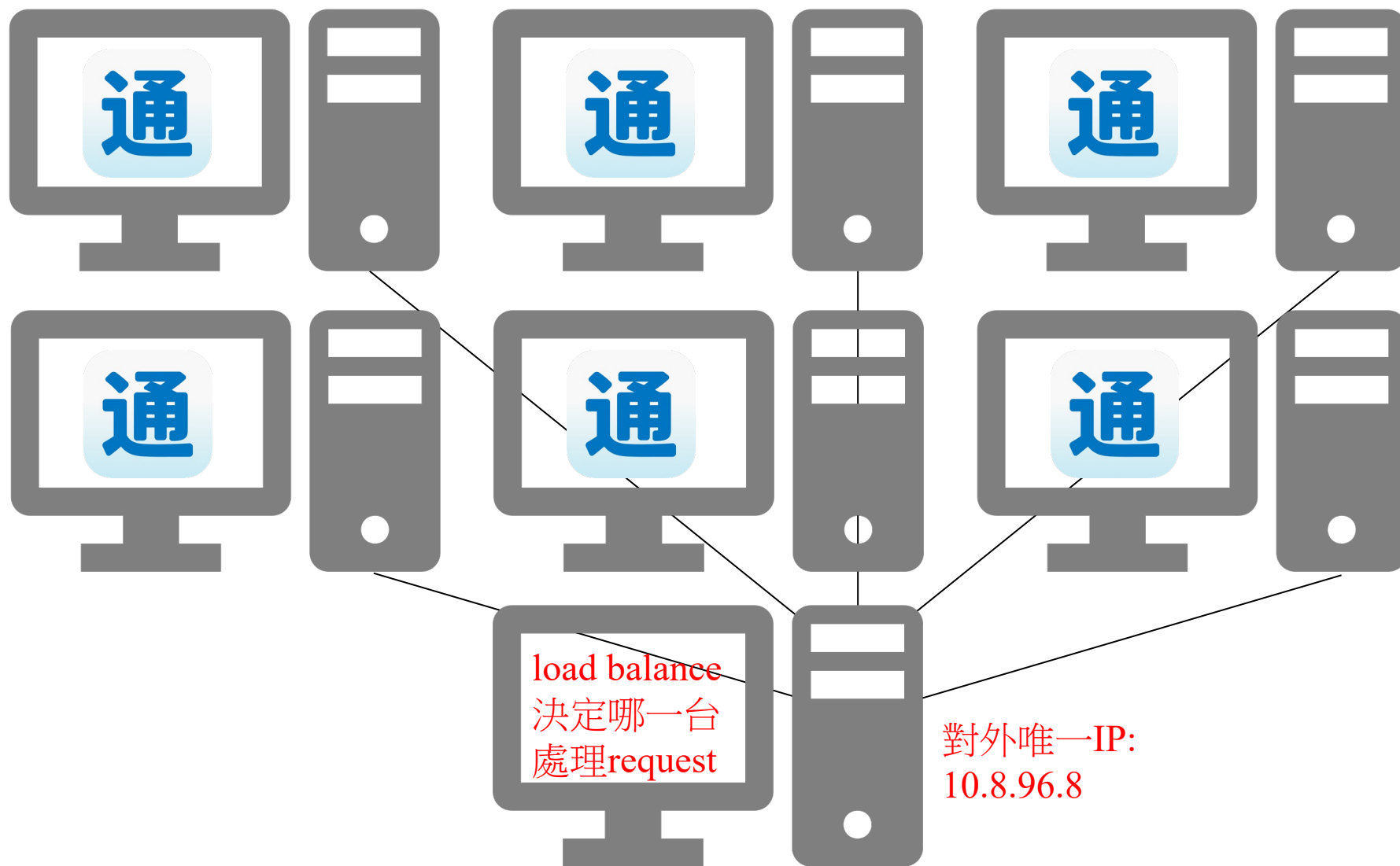
小明是個工程師，做出了大大通網站



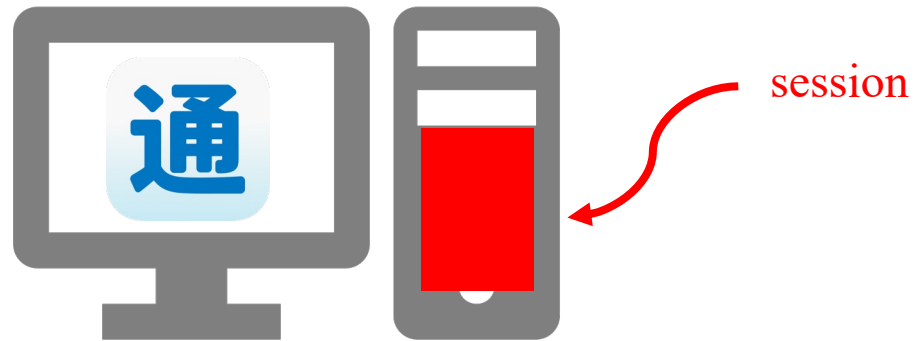
好忙... 找幾個部下好了



突然跑不動了... 加爆！

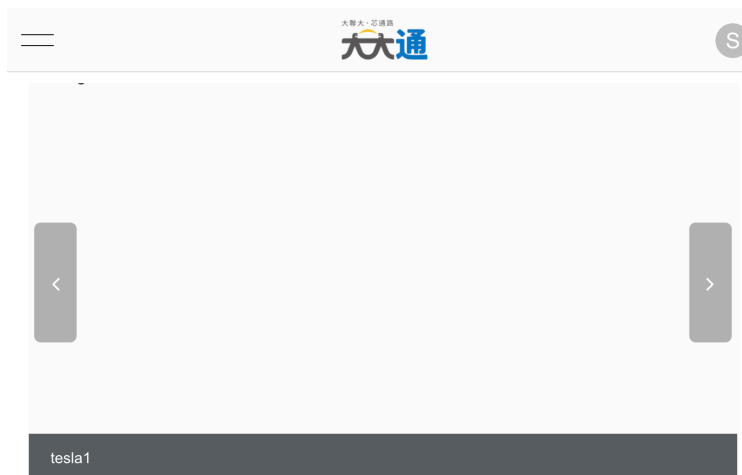


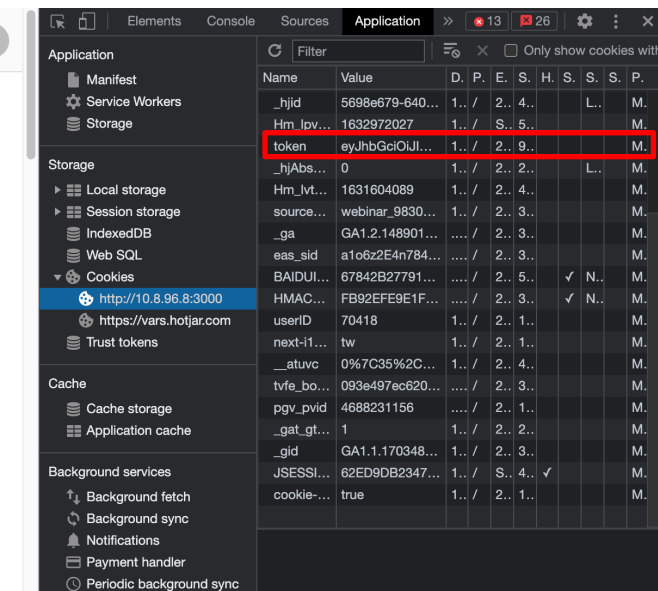
# 偷偷塞



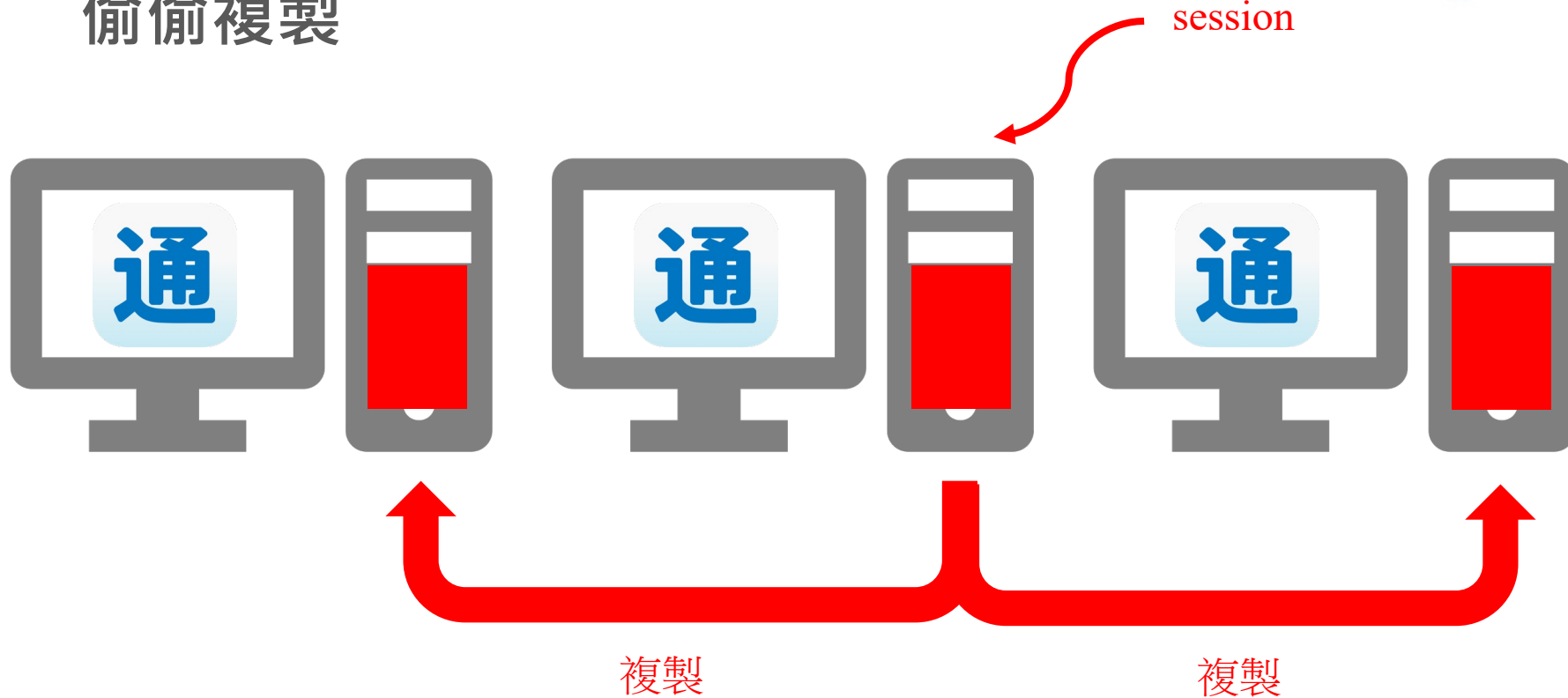
# 在 RAM 裡的 Session

| key          | value   |
|--------------|---|
| eyJhbGciOiJi | {<br>username: Samuel,<br>email: <u>samuel.luo@wpgholdings.com</u><br>} |
| a432gre1qfel | {...}   |
| 902hg8412pfo | {...}   |



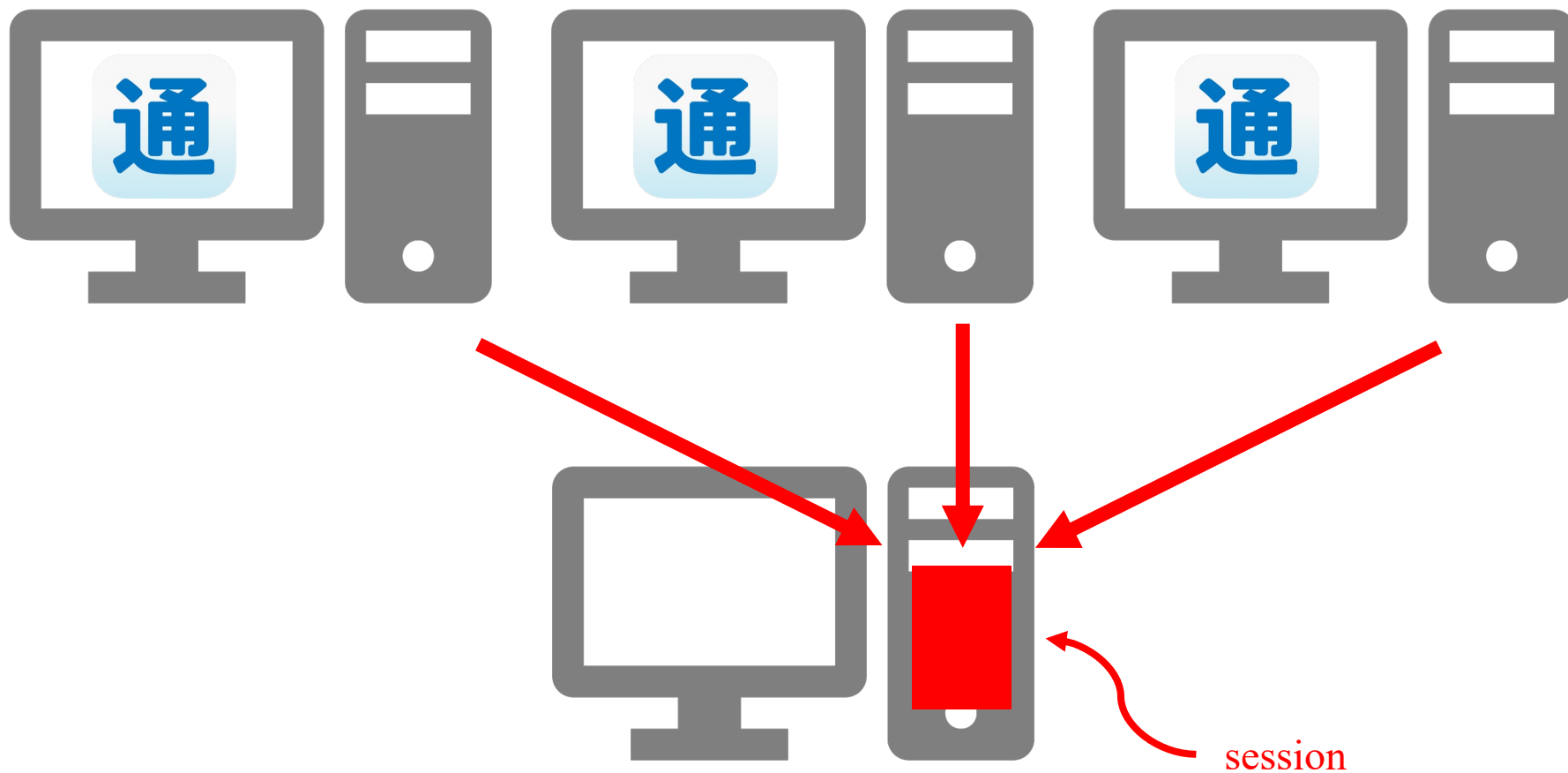


# 偷偷複製





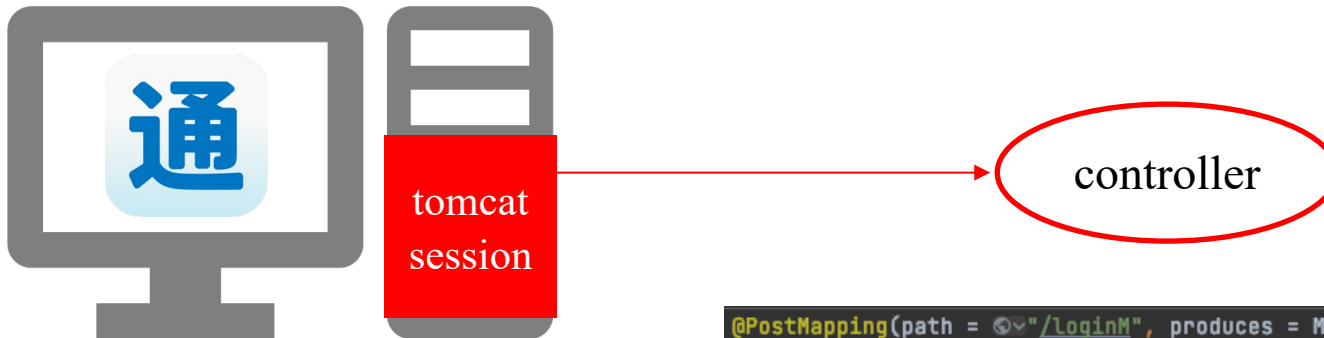
另外一台專門放session



有狀態 (Stateful) -> 無狀態 (Stateless)

# Session in Spring

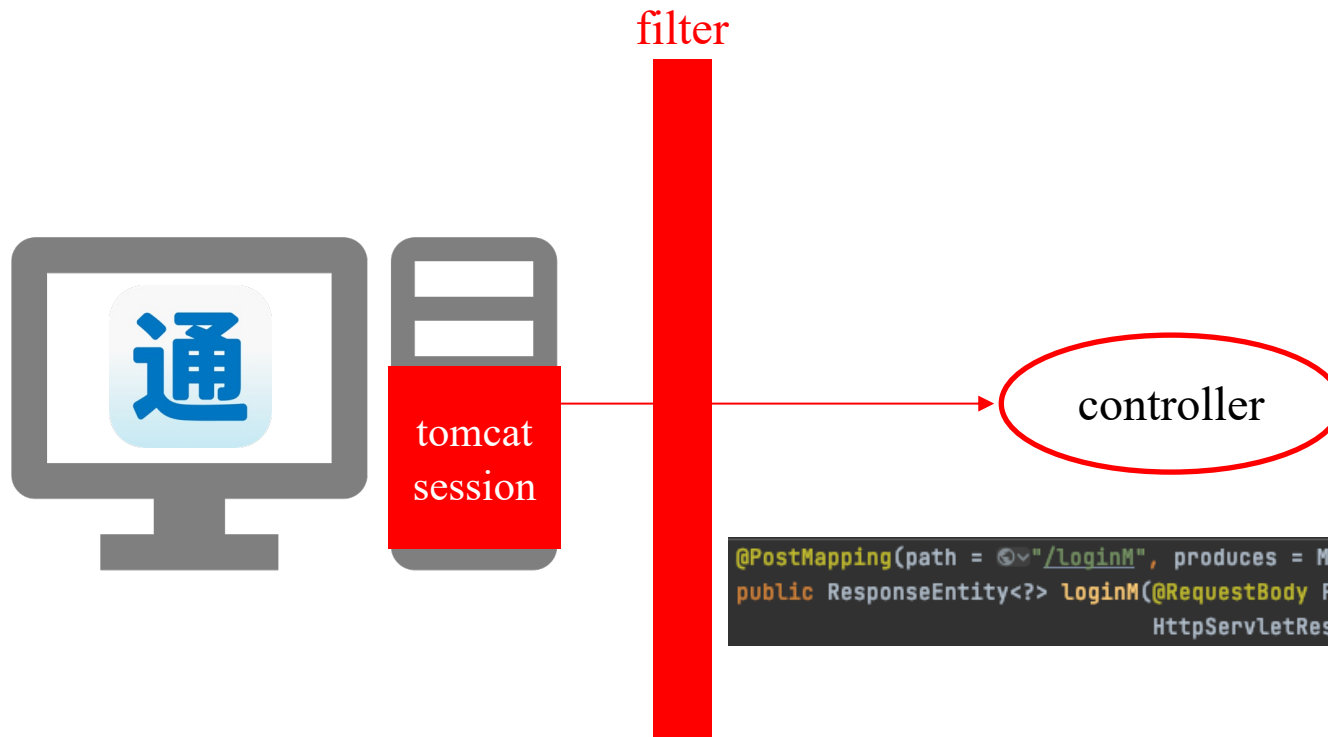
session 跟 Tomcat 綁在一起



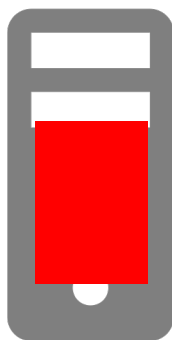
```
@PostMapping(path = "/loginM", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<?> loginM(@RequestBody PaasModel paasModel, HttpServletRequest req,
                                HttpServletResponse httpResponse, HttpSession session) {
```

# Redis Session in Spring

加入 dependency: spring-session-data-redis



# 回到剛剛的故事



用戶太愛提問了...

提問功能



提問以外的功能

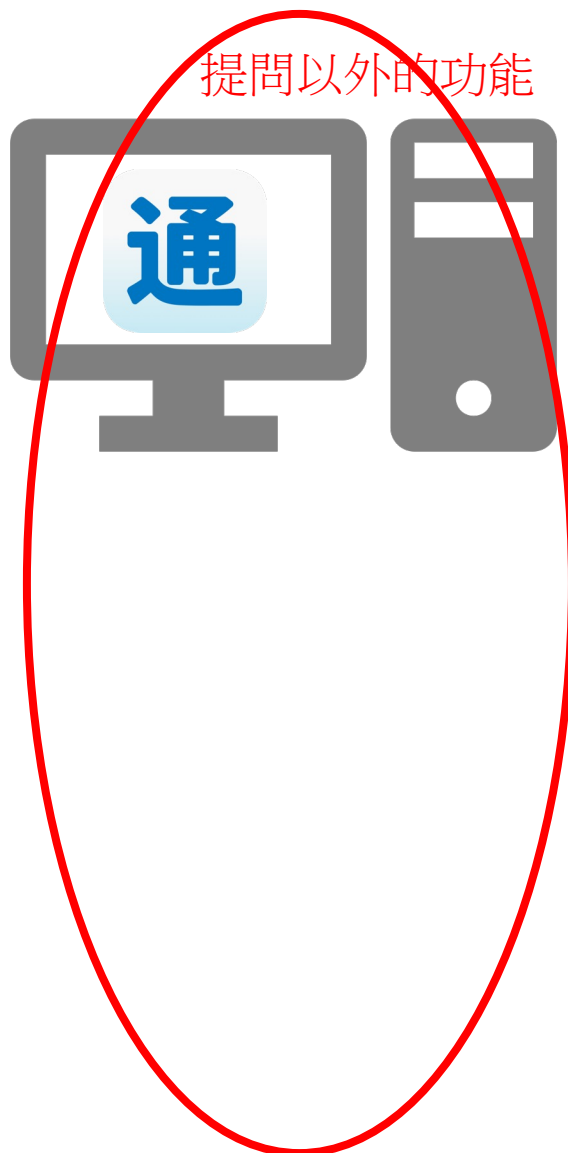


# 不互相影響

提問功能

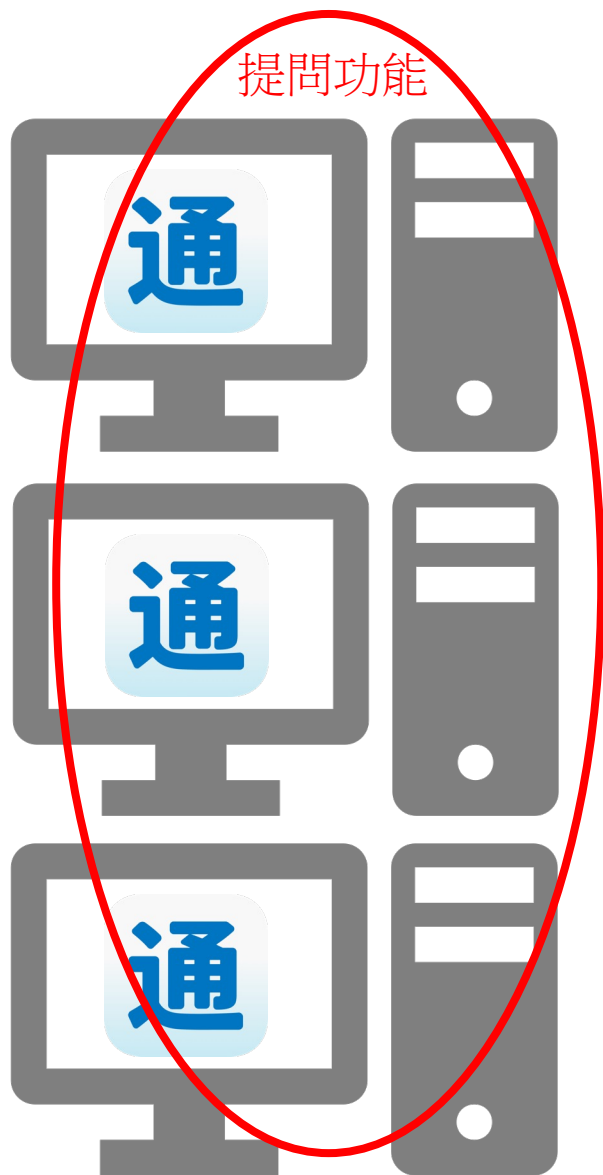


提問以外的功能

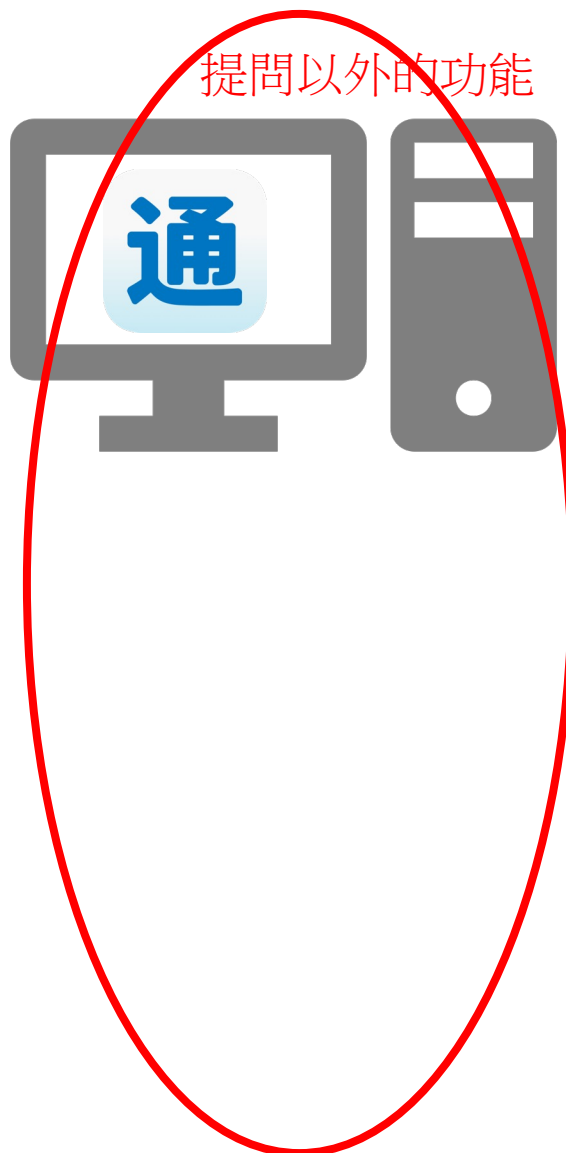


# 後台管理自己人使用

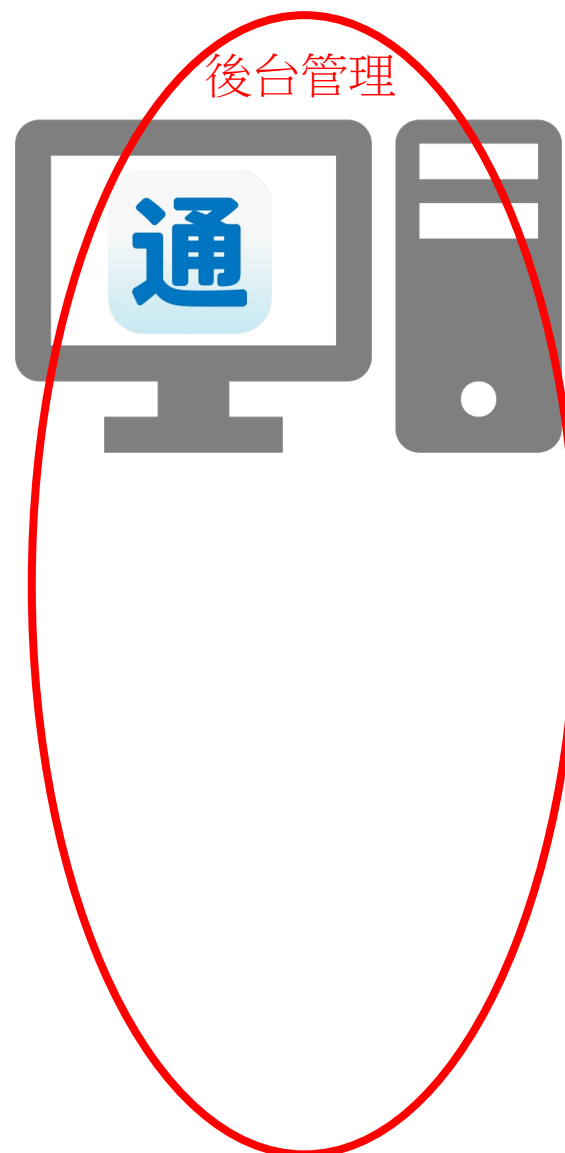
提問功能



提問以外的功能

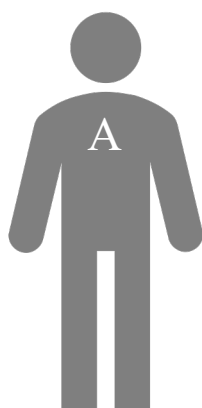


後台管理

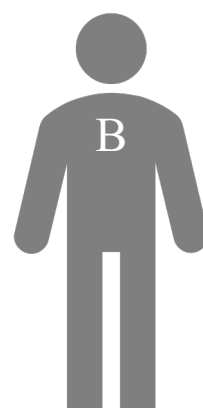


# 各自負責各系統 – 微服務 (Microservice)

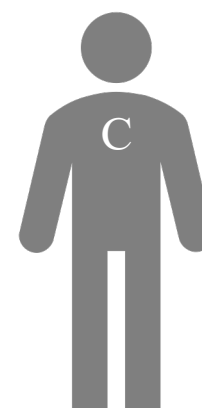
提問功能



提問以外的功能



後台管理

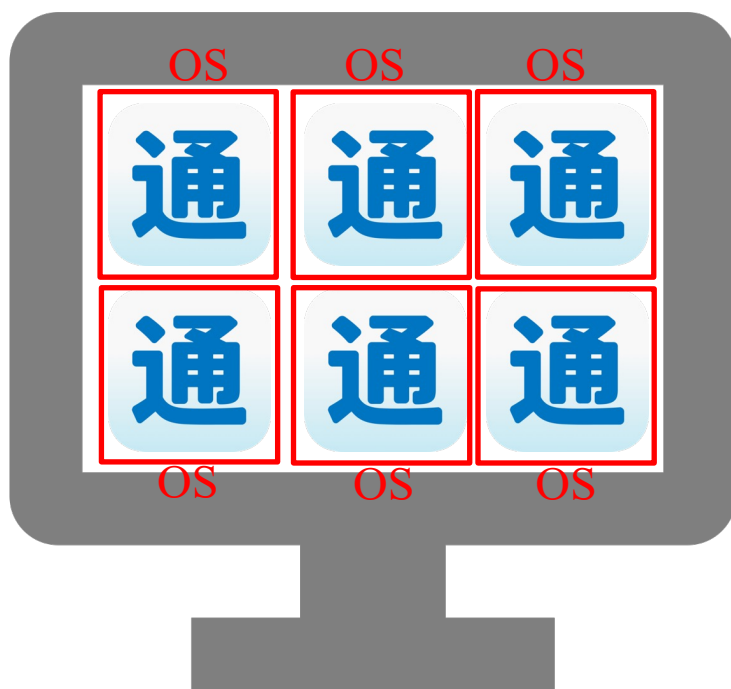
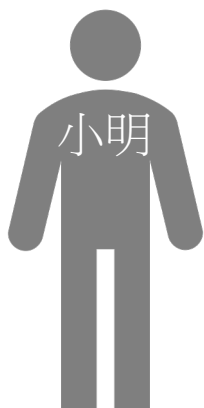




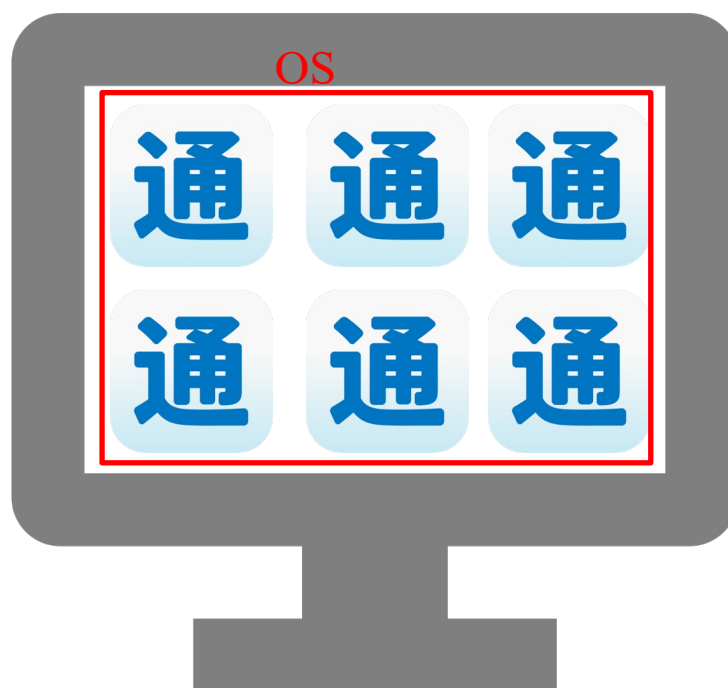
RAM和CPU都沒用光，好浪費！



# 一台伺服器給我裝一大堆OS – 虛擬機器(VM)



# 全部運行在同一個OS



# 一大堆系統運行在同一OS...會有問題吧？

1. A系統要用JAVA 8 以下、B系統要用JAVA 9 以上 (衝突了！)

-> Linux有提供pivot\_root api

-> 每個系統都在獨立的環境——下載自己的依賴套件

/usr/user1/project1/... 變成 /...

/usr/user1/project2/.... 變成 /...

# 一大堆系統運行在同一OS...會有問題吧？

2. 某一process可以kill掉另外一個process

3. 每個系統要有自己的hostname

4. 佔用的port衝突

-> Linux有提供namespace api

# 一大堆系統運行在同一OS...會有問題吧？

5.某一個系統佔用了一大堆ram和cpu，害到其它系統

-> Linux有提供cgroups api

# Docker

寫腳本call 這些linux api?

-> 交給Docker

-> 寫Dockerfile告訴Docker該怎麼 build 環境、要怎麼執行起來系統

-> 這些透過Docker執行起來的系統都叫容器(container)

# 大大通實際作法

下載java -> 宣告JAVA\_HOME -> 下載gradle -> 宣告gradle home

```
FROM gradle:5.1.1-jdk8-alpine AS TMP_BUILD_IMAGE
ARG ROOT_PATH=/WPG/sources/dadawant
ARG DDT_RESOURCES_PATH=/WPG/sources/dadawant/dadatong/src/main/resources

COPY --chown=gradle:gradle / $ROOT_PATH/dadatong/

USER gradle
RUN mv -f $ROOT_PATH/dadatong/build.gradle.test $ROOT_PATH/dadatong/build.gradle \
    && mv -f $DDT_RESOURCES_PATH/sqs.properties.docker $DDT_RESOURCES_PATH/sqs.properties
WORKDIR $ROOT_PATH/dadatong
RUN gradle createWar
```

運用gradle build 出 WAR



# 大大通實際作法

`FROM tomcat:9.0.12-jre8-slim` 下載java -> 宣告JAVA\_HOME -> 下載tomcat -> 宣告catalina home

`ARG ROOT_PATH=/WPG/sources`

`ARG DDT_PATH=/WPG/sources/dadawant/dadatong`

`ARG DDT_RESOURCES_PATH=/WPG/sources/dadawant/dadatong/src/main/resources`

`COPY src/main/resources $DDT_RESOURCES_PATH`

準備好該環境所需的文件和資料夾

`RUN mv -f /usr/local/tomcat/ $ROOT_PATH/tomcat/ \`

`&& rm -rf $ROOT_PATH/tomcat/webapps/* \`

`&& mv -f $DDT_RESOURCES_PATH/sqs.properties.docker $DDT_RESOURCES_PATH/sqs.properties \`

`&& mv -f $DDT_RESOURCES_PATH/DBLink.properties.test $DDT_RESOURCES_PATH/DBLink.properties \`

`&& mv -f $DDT_RESOURCES_PATH/log4j.properties.template $DDT_RESOURCES_PATH/log4j.properties \`

`&& mkdir -p $DDT_PATH/D: \`

`&& mkdir -p $DDT_PATH/upload \`

`&& mkdir -p $DDT_PATH/logForDsu/AllLog`

`COPY --from=TMP_BUILD_IMAGE $DDT_PATH/build/libs/ROOT.war $ROOT_PATH/tomcat/webapps`

從剛剛那個stage複製WAR檔過來

# 大大通實際作法

```
RUN useradd -ms /bin/bash dadatong
WORKDIR $ROOT_PATH
    使用者從 root 切換至 dadatong
COPY docker/entrypoint.sh entrypoint.sh
RUN chmod -R 777 $ROOT_PATH \
    && chown dadatong:dadatong entrypoint.sh
USER dadatong
```

```
EXPOSE 8080
```

系統run起來

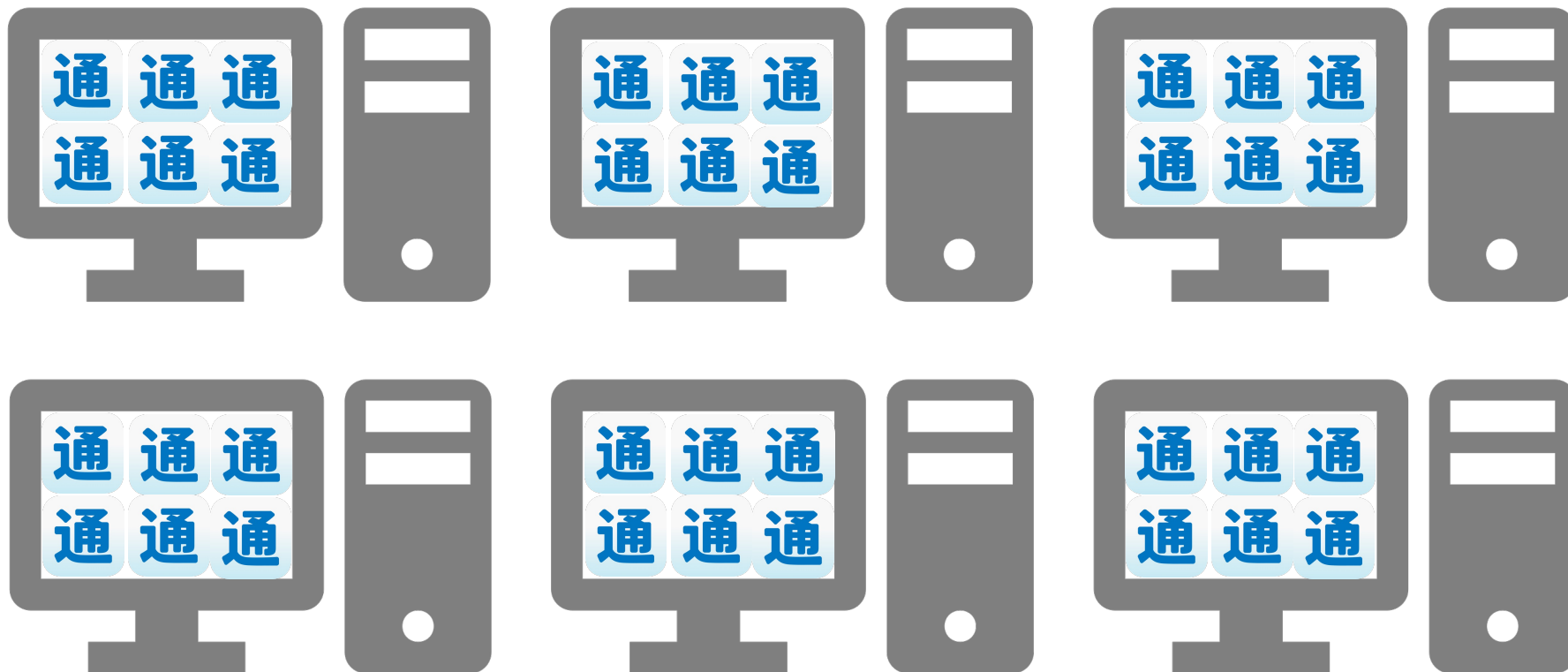
```
ENTRYPOINT ["./entrypoint.sh"]
```

```
#!/bin/bash
```

entrypoint.sh

```
export ROOT_PATH=/WPG/sources/dadawant/dadat
export CATALINA_HOME=/WPG/sources/tomcat
export PATH=${JAVA_HOME}/bin:${PATH}
export JAVA_OPTS="-Dfile.encoding=UTF-8 -Dpr
${CATALINA_HOME}/bin/catalina.sh run -config
```

頭好痛...



# 管理這些有好多問題

1. 有一個service要用新的版本
  2. 新的版本有問題 -> 用舊的版本
  3. 某一個service快爆了 -> 多幾個容器來處理request
  4. load balance指定要哪一個容器處理request
  5. 定時監控所有容器，某個容器掛了，再起一個
  - ...
- > 用Kubernetes就好!

# Kubernetes(K8s)

- 1.特別適合無狀態、微服務的系統
- 2.多開一台機器掌控所有容器
- 3.管理容器們的解決方案

**Q & A**



*Thank you*

產業首選 · 通路標竿