Practica 4 – Benchmarking y Ajustes del Sistemas

Benchmark: Conjunto de test que nos permiten medir el sistema

Testear CPU o RAM, HDD y SDD no aconsejable por peligro hacia sectores del disco.

Saber de Phoronix:
Instalación, **phoronix-test-suite list-available-test** y **phoronix-test-suite benchmark <TEST NAME>**, phoromatic, funciona a nivel de sistema

Saber ab a nivel de servicio HTTP, uso:
ab [-m customizar testeo sobre HTTP] [–n nº peticiones] [–c número de peticiones múltiples por unidad de tiempo] <IP><Path>

No hacemos test desde Ubuntu a CentOS y viceversa porque el sistema que envía también influye en la métrica, por tanto se debe hacer desde una máquina externa. Tampoco se hace a una máquina a sí misma porque el sistema (y variables que no queremos medir) influirían en el benchmark. Estaríamos midiendo la capacidad de proporcionar servicio a la vez que solicitarlo. Desde el host es la mejor opción que tenemos.

GET (Solicitar HTML) – POST (Solicitar modificación/añadir en el servidor) – PUT – DELETE
Existe un servidor Node JS y otro MongoDB. docker_compose (que usa una herramienta de microservicios llamada docker) enlaza ambos para usarlos en la app.

Los contenedores tienen ventajas sobre el uso de recursos y la reutilización (frente a las VM), ofrecen independencia frente al SO y facilidad de plug-and-play.

A través de JMeter simular peticiones de usuarios (recibirán un token/código(JWT) de autenticación que permiten identificarlos y mostrarle recursos distintos). El programa hace un POST, recibe un token y luego solicita un GET.

Docker tiene dos conceptos claves: images (similar a una ISO ligera) y container (una image en ejecución, ver en ps)
Si no tiene la imagen se puede hacer: **docker pull <nombre>** y lanzarla con **docker run <nombre>**

- Instalar Docker siguiendo la página de DigitalOcean y PDF (genera dockers aislados)
- Instalar Docker Compose (Ansible de contenedores): Lanzará un contenedor MongoDB y uno NodeJS y los enlazará (el script mapea los puertos “virtuales” a los de la máquina, los expone)
- En el directorio donde está instalado el archivo yml nos situamos y ejecutamos **docker-compose up** (descarga las imágenes y monta el servidor), el servidor recibe las peticiones en el puerto 3000
- Desde el navegador **192.168.56.105:3000**, desde la terminal vemos el funcionamiento
    - Hay un script para hacer las operaciones: **./pruebaEntorno.sh**
        - También se puede hacer desde PostMan (app de Chrome ¿?)
        - Devuelve una cadena que se decodifica en jwt.io (realmente es un token usado para el GET)
        - Pegar token en Headers – Bearer (o value), devolviendo un JSON
    - Los POST van a 192.168.56.105:3000/api/v1/auth/login
    - En la carpeta jMeter hay una lista de usuarios/administradores con sus contraseñas en formato csv.

<u>Instalación y uso de Phoronix</u>

- Instalamos
  sudo apt-get install phoronix-test-suite

  (Es posible que haya que actualizar con **sudo apt-get update**)

Navegador necesario
sudo apt-get install firefox

En **/etc/ssh/sshd_config** permitimos el acceso mediante X11 (viene hecho)

Instalamos módulo PHP
sudo apt-get install php-zip

- Nos conectamos por SSH
  ssh –X –p 22022 ive@192.168.56.105 (-X para poder acceder a interfaz gráfica)

- Ejecutamos Phoronix
  phoronix-test-suite gui

Pasos Docker

- Instalamos Docker
  Añadimos llave GPG para validar el repositorio
  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

  Añadimos repositorio
  sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

  Actualizamos lista de repositorios
  sudo apt update

  Instalamos el repositorio de docker
  sudo apt install docker-ce

  (se puede comprobar con **sudo systemctl status docker**)

  Añadimos el usuario al grupo docker
  sudo usermod –aG docker ive

  (Debemos salir con **exit** y volver a entrar, ahora no hace falta usar sudo para los comandos de docker)
  (Se puede probar con **docker info** o con **docker run hello-world**)

- Instalamos Docker Compose
  sudo apt install docker-compose

  (Se puede probar con **docker-compose --version**)

- Instalamos la app
  Clonamos desde git
  git clone https://github.com/davidPalomar-ugr/iseP4JMeter.git

  Accedemos al directorio y lanzamos la aplicación
  cd iseP4JMeter
  docker-compose up

Pasos JMeter

- Creamos un Test Plan con dos Thread Group y un Test Fragment
- Definimos variables globales para la IP (HOST – 192.168.56.105) y para el puerto (PORT - 3000). Se acceden con ${…}
- Creamos un HTTP Request Defaults con acceso a **/api/v1**, también un HTTP Authorization Manager con la URL, el usuario y la contraseña.
- Indicamos en los **Thread Group** el número de hebras, el tiempo en el que deben estar lanzadas y el número de repeticiones.
- Para cada **Thread Group** añadimos un **CSV Data Set Config** indicando el nombre de las variables, que se ignore la primera línea, el archivo del que se coge y el delimitador (en nuestro caso la coma).
- Creamos 3 **HTTP Request**, 2 para el login de los alumnos y los administradores (indicando el path y los parámetros enviados (se cogen de las variables de CSV)) y otro para solicitar la información de un alumno.
- En los Login se añade un extractor de token con expresión regular **.+** y template **0**, se define variable **token**.
- Se añaden **Gaussian Random Timer**
- Se crea **Access Log Sampler** que vuelca en **apiAlumnos.log**, sirva para muestrear el acceso de administradores.
- Un **HTTP Header Manager** con **Authorization Bearer $(token)** a la recuperación de los datos del alumno y al acceso de los administradores.
- Se introducen los siguientes **Listeners**: **Summary Report, View Results in Table, Aggregate Report**.

Comandos	
docker-compose up	Lanzar app
phoronix-test-suite gui	Lanzar interfaz Phoronix
phoronix-test-suite list-available-test	Lista los distintos commandos con los que se puede lanzar Phoronix
phoronix-test-suite benchmark <TEST NAME>	Lanza un benchmark en concreto
ab [-m GET/POST…] [–n nº peticiones] [–c número de peticiones múltiples por unidad de tiempo] <IP> <Path>	Ejecución de AB

Tipo de preguntas de examen:

- Docker crea un contenedor para Mongo y otro para Node y los conecta, siguiendo las instrucciones de un script **yml**.
- **¿Por qué no se hacen tests usando únicamente las máquinas virtuales?**
  No hacemos test desde Ubuntu a CentOS y viceversa porque el sistema que envía también influye en la métrica, por tanto se debe hacer desde una máquina externa. Tampoco se hace a una máquina a sí misma porque el sistema (y variables que no queremos medir) influirían en el benchmark. Estaríamos midiendo la capacidad de proporcionar servicio a la vez que solicitarlo. Desde el host es la mejor opción que tenemos.
- Hay autenticación a la hora de obtener el token por seguridad, para que no todo el mundo pueda saturar al servidor solicitando tokens.