CREANDO SOLUCIONES DE CALIDAD

Miguel Ángel Alameda

Criterios de aceptación

- **Código fuente en GIT**

  Disponible en https://gitlab.keepcoding.io/malameda/soluciones-calidad

- **Sistema con más de dos microservicios**

El sistema montado cuenta con tres microservicios, un frontend y dos backends para gestionar usuarios y grupos.

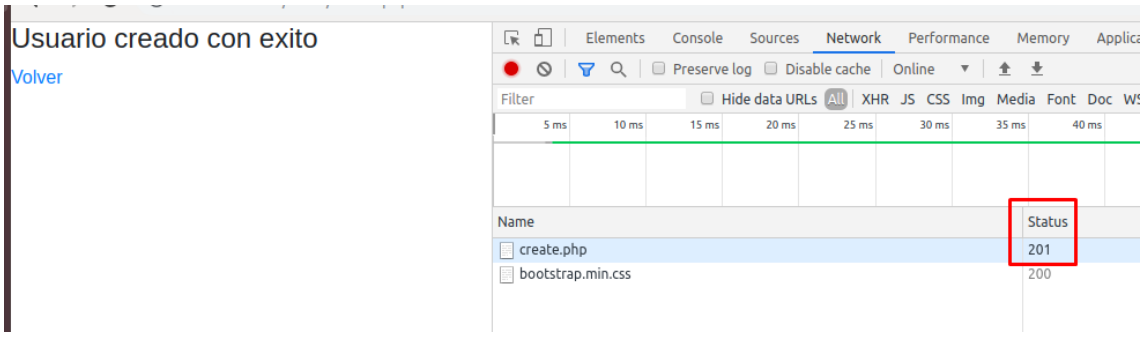
- **Comunicación entre servicios por HTTP, RPC o Colas.**

La comunicación entre microservicios se realiza a través de llamadas y respuestas HTTP:

```php
$URL="http://web-back-users/index.php";

    $toSend = new stdClass();
    $toSend->name= $_POST['create_name'];
    $toSend->email= $_POST['create_email'];
    $jsonToSend = json_encode($toSend);

    $ch = curl_init();
    curl_setopt($ch,CURLOPT_URL,$URL);
    curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
    curl_setopt($ch,CURLOPT_POST, 1);
    curl_setopt($ch,CURLOPT_POSTFIELDS,$jsonToSend);
    $respuesta = curl_exec($ch);
```



- **Manifest files de Kubernetes para desplegar el sistema con “kubectl apply -f .”.**

Dentro del proyecto, disponemos del directorio “k8s”. Aquí contamos con todos los ficheros .yaml necesarios para desplegar la aplicación en Kubernetes.

- **Se puede navegar en el sistema con una interfaz web (UI o API).**

La aplicación dispone de una pequeña interfaz de usuario para navegar:

- **Incluye instrucciones para correr y usar la aplicación (comando y URL de prueba).**

Disponibles en el fichero readme.md del proyecto.

- **No tiene configuraciones quemadas en el código sino usa variables de ambiente.**

Se utilizan variables de entorno para establecer la conexión con BBDD, usando el mismo fichero de secretos para la configuración del POD de mysql y para establecer las variables que se usarán después en la aplicación, para ambos microservicios backend (los que disponen de BBDD). Las variables de entorno se encuentran en los “deployment” de PHP para ambos microservicios backend:

```
        ports:
        - containerPort: 9000
          name: php-back-groups
        env:
          - name: MYSQLG_DATABASE
            valueFrom:
              secretKeyRef:
                name: mysql-users-env
                key: MYSQL_DATABASE
          - name: MYSQLG_USER
            valueFrom:
              secretKeyRef:
                name: mysql-users-env
                key: MYSQL_USER
          - name: MYSQLG_HOST
            value: "mysql-back-groups"
          - name: MYSQLG_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mysql-users-env
                key: MYSQL_PASSWORD
          - name: DBG_SERVER_TYPE
            value: "mysql"
```

Posteriormente la conexión a BBDD se establece así:

```
public function connect()
{
        if(self::$connection_status == null)
    try{
      self::$connection_status = new PDO($_ENV['DBG_SERVER_TYPE'].':host='.$_ENV['MYSQLG_HOST'].';dbname='.$_ENV['MYSQLG_DATABASE'].'', $_ENV['MYSQLG_USER'],
    }catch(PDOException $e)
    {
        die($e->getMessage());
    }
    return self::$connection_status;
}
```

- **Se puede habilitar/deshabilitar alguna feature fácilmente (incluye instrucciones).**

A través de una variable de entorno definida en el deployment del servicio PHP en el frontend, se puede visualiza o no la eliminación de usuarios y grupos en el formulario:

```
    spec:
      containers:
      - image: mmiguel80/php-frontend:v1.1
        name: php-frontend
        resources:
          requests:
            memory: "64Mi"
            cpu: "80m"
          limits:
            memory: "256Mi"
            cpu: "250m"
        ports:
        - containerPort: 9000
          name: php-frontend
        env:
          - name: APP_ENV
            value: "dev"
```

Si la variable de entorno “APP_DEV” es igual a “dev”, visualizaremos la opción en el formulario. En caso contrario, no se visualizará:

```
<!-- Bloque eliminar -->
<?php
if (isset($_ENV['APP_ENV']) && $_ENV['APP_ENV'] == "dev"){
?>
<div class="block">
    <div class="right">
        <form action="groups/delete.php" method="post">
            <h3 for="exampleInputEmail1">Eliminar grupo (dev)</h3>
            <div class="form-group">
                <input type="text" class="form-control" name="delete_id"
            </div>
            <div class="form-group">
                <input type="submit" name="delete" value="Enviar" class="
            </div>
        </form>
    </div>
    <div class="left">
        <form action="users/delete.php" method="post">
```

- **Cada microservicio cuenta con su propia base de datos o almacenamiento de estado.**

Los dos microservicios backend disponen de una BBDD MySql cada uno, que gestionan por separado.

- **Un microservicio consulta las base de datos de otro servicio indirectamente como un API.**

El microservicio frontend es el que consulta a los dos backends para gestionar usuarios/grupos. Como se ha mostrado anteriormente, se hacen peticiones a los servicios disponibles en cada backend y ellos gestionan la conexión a BBDD para devolver la info solicitada.

- **Se puede observar la aplicación desde afuera. Es decir, emite logs, traces, metrics.**

He utilizado stackdriver para ver logs y trazas de la aplicación:

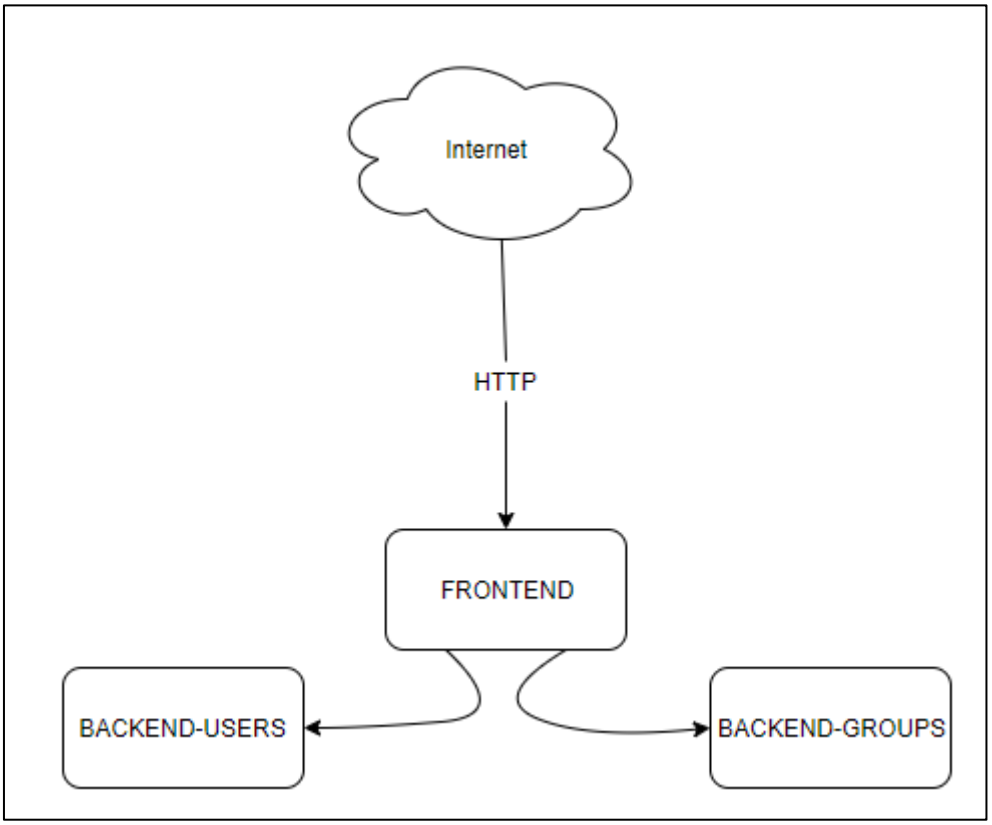
```
2019-10-27 21:32:50.231 CET  10.4.3.9 - 27/Oct/2019:20:32:50 +0000 "GET /index.php" 200
{
  insertId: "mgm8r1xpgmbu0zm3p"
  labels: {
    k8s-pod/app: "back-users"
    k8s-pod/pod-template-hash: "7fd6fc9db4"
    k8s-pod/tier: "php-back-users"
  }
  logName: "projects/swift-branch-257217/logs/stderr"
  receiveTimestamp: "2019-10-27T20:32:56.628564660Z"
  resource: {…}
  severity: "ERROR"
  textPayload: "10.4.3.9 -  27/Oct/2019:20:32:50 +0000 "GET /index.php" 200
"
  timestamp: "2019-10-27T20:32:50.231017403Z"
}
```

- **Incluye proyecto de pruebas de algun tipo (unitarias, integracion, carga, etc.).**

Se han includo pruebas con lint y con PHPUnit en los dos backend (simulando un test unitario). Estos tests se realizan en la construcción del contendor(multi-stage), lo que nos asegura que cuando se construye una imagen, ya ha pasado los test necesarios.

- **Incluye diagrama de las dependencias de todos los servicios (mapa de Istio es válido).**

  Al tener sólo 3 microservicios, el mapa es muy sencillo. También se encuentra en el readme.md



- **Incluye evidencias claras de al menos cinco buenas prácticas que se vieron durante el módulo (capturas de pantalla en código)**

Automatización: Pasamos los test en la construcción de la imagen:

```
1  FROM jakzal/phpqa:alpine
2  COPY ./www /project/
3  COPY ./test /project/
4  WORKDIR /project
5  RUN parallel-lint index.php
6  RUN parallel-lint lib/database.php
7  RUN parallel-lint lib/functions.php
8  RUN phpunit DatabaseTest.php
9
10
11 FROM php:7.2.7-fpm
12 RUN docker-php-ext-install mysqli pdo_mysql
13 COPY ./www /var/www/html/
```

Formatos de logs de Apache normalizados para stackdriver (logentry):

```
  # Send apache logs to stdout and stderr
  LogFormat "{ \"time\":\"%t\", \"remoteIP\":\"%a\", \"host\":\"%V\", \"request\":\"%U\", \"query\
  CustomLog /proc/self/fd/1 logentry_apache
  ErrorLogFormat "{ \"time\":\"%{%Y-%m-%d}tT%{%T}t.%{msec_frac}tZ\", \"function\" : \"[%-m:%l]\" ,
  ErrorLog /proc/self/fd/2
/VirtualHost>
```

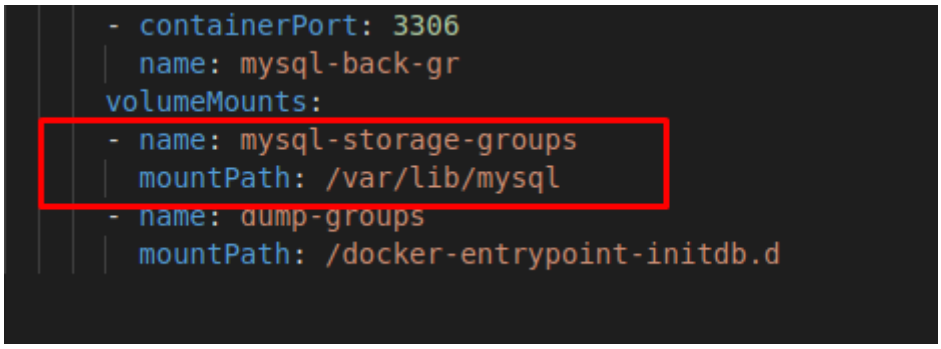
Construir, Distribuir, Ejecutar:

Fácil hacer rollback cambiando a una versión anterior la imagen de Docker que se utiliza en el despliegue:

```
      tier: php-back-groups
  spec:
    containers:
    - image: mmiguel80/php-back-groups:v1.0
      name: php-back-groups
      resources:
        requests:
          memory: "64Mi"
```

Para desplegar, utilizamos el paquete (imagen Docker en Docker Hub) y la configuración (manifest files).

Persistencia en el backing service:



Asignación de puertos:

- Servicio publicado mediante asignación de puertos
- Aplicación auto-contenida.
- Service- Discovery (svc de K8s).

Concurrencia:

Se puede escalar horizontalmente mediante modelo de procesos.

Deshechabilidad:

Escalado rápido y flexible

Procesos desechables.

Podemos ir eliminando réplicas poco a poco.

Escalado horizontal.

Logs:

Log en salida estándar.

Se visualizan además en StackDriver.

No gestiona archivos.

Observabilidad desde stackdriver:

Se utiliza una **feature flag** para mostrar una parte del código (simulando entorno DEV).