m4gm.com /moodle/course/view.php

Curso: Desarrollo de Sistemas Distribuidos

92-117 minutes

Su progreso

Esquema de tópicos/temas

- General

General

- El examen es individual y se deberá presentar en forma remota utilizando la plataforma Moodle.

  El examen se podrá presentar el lunes 16 de junio a las 16:30 Hrs. con una duración máxima de 90 minutos.

  Los temas a evaluar son los siguientes:

  - Servicios de nombres, archivos y replicación.
    - Nombre, identificador y dirección.
    - Espacios de nombres.
    - Árbol de nombres.
    - Grafo dirigido no cíclico.
    - Resolución de nombres.
    - Mecanismo de clausura.
    - Vinculo absoluto y vinculo simbólico.
    - Montar un espacios de nombres.
    - El sistema de archivos distribuidos NFS.
    - Domain Name System (DNS).
    - Capa global.
    - Capa de administración.
    - Capa de dirección.
    - Resolución de nombre en un DNS.
    - Resolución iterativa.
    - Resolución recursiva.
    - Instalación en el servidor.
    - Instalación en el cliente.
    - Probar el acceso a archivos remotos.
    - ¿Por qué replicar los datos?
    - Modelos de consistencia.
    - Consistencia secuencial.
    - Consistencia de entrada.
    - Respaldos incrementales.
    - Respaldos continuos.
    - Replicación del sistema completo.
    - Arquitectura de un sistema replicado.
    - Azure Site Recovery.
  - Cómputo en la nube.
    - La elasticidad en la nube.
    - Nube pública, nube privada y nube híbrida.

- Escenario On-premise.
- Escenario nube.
- Cambio del tamaño de una máquina virtual.
- Cambio del tamaño del disco de S.O.
- Microsoft Azure Backup.
- Habilitar el respaldo de una máquina virtual en Azure.
- Iniciar un respaldo completo.
- Restaurar una máquina virtual.
- Eliminar un proceso de respaldo.
- Crear la imagen de una máquina virtual.
- Crear una máquina virtual a partir de una imagen.
- Azure Site Recovery.
- Crear la máquina virtual.
- Replicar una máquina virtual.
- Azure Load Balancer.
- Balanceador de carga público.
- Balanceador de carga interno.
- Escalamiento mediante balance de carga.
- Costo del balance de carga en Azure.
- Creación de un balanceador de carga en Azure.
- Configuración del balanceador de carga.
- Agregar un sondeo de estado.
- Agregar una regla de equilibrio de carga.
- Agregar una regla NAT de entrada.
- Platform as a Service.
- Azure Database for MySQL.
- Creación de un servidor en Azure Database for MySQL.
- Conexión al servidor MySQL.
- Software as a Service.
- ¿En qué consiste el servicio de SendGrid?
- Envío de email utilizando el API de SendGrid.
- Reportes en tiempo real.

- 4. Servicios de nombres, archivos y replicación

4. Servicios de nombres, archivos y replicación

- Desarrollar un programa en Java que implemente un chat utilizando comunicación multicast mediante datagramas.

  Se **deberá** ejecutar el programa en una máquina virtual con Windows 10 en Azure. Solo se admitirá la tarea si se trata de un programa en modo consola de caracteres (no se admitirá el programa en modo gráfico).

  Se **deberá** pasar como parámetro al programa el nombre del usuario que va escribir en el chat. Para demostrar el programa se **deberá** utilizar los siguientes usuarios: hugo, paco y luis (no usar otros usuarios).

  El programa **deberá** utilizar las siguientes funciones para enviar y recibir los mensajes multicast:

```
static void envia_mensaje_multicast(byte[] buffer,String ip,int puerto)
throws IOException
{
  DatagramSocket socket = new DatagramSocket();
  socket.send(new
DatagramPacket(buffer,buffer.length,InetAddress.getByName(ip),puerto));
  socket.close();
}

static byte[] recibe_mensaje_multicast(MulticastSocket socket,int
longitud_mensaje) throws IOException
{
  byte[] buffer = new byte[longitud_mensaje];
  DatagramPacket paquete = new DatagramPacket(buffer,buffer.length);
```

```
  socket.receive(paquete);
  return paquete.getData();
}
```

El funcionamiento general del programa es el siguiente:

- El programa creará un thread que actuará como cliente multicast, el cual recibirá los mensajes del resto de los nodos. Cada mensaje recibido será desplegado en la pantalla. El thread desplegará el mensaje que envía el mismo nodo.
- En el método main(), dentro de un ciclo infinito se desplegará el siguiente prompt: "**Ingrese el mensaje a enviar: **" (sin las comillas), entonces se leerá una string (el mensaje). Se **deberá** enviar el mensaje a los nodos que pertenecen al grupo identificado por la IP 230.0.0.0 a través del puerto 50000. **El paquete a enviar deberá tener la siguiente forma**: *nombre_usuario***:***mensaje_ingresado*, dónde *nombre_usuario* es el nombre del usuario que pasó como parámetro al programa (hugo, paco o luis) y *mensaje_ingresado* el mensaje que el usuario ingresó por el teclado.

Se **deberá** completar el siguiente programa:

```
class Chat
{
  static class Worker extends Thread
  {
    public void run()
    {
     // En un ciclo infinito se recibirán los mensajes enviados al grupo
     // 230.0.0.0 a través del puerto 50000 y se desplegarán en la
pantalla.
    }
  }
  public static void main(String[] args) throws Exception
  {
    Worker w = new Worker();
    w.start();

    String nombre = args[0];

    // En un ciclo infinito se leerá cada mensaje del teclado y se
enviará el mensaje al
    // grupo 230.0.0.0 a través del puerto 50000.
  }
}
```

Para probar el programa, se **deberá** ejecutar la siguiente conversación en tres ventanas de comandos (cmd) en la máquina virtual con Windows 10. En la primera ventana escribirá hugo, en la segunda ventana escribirá paco y en la tercera ventana escribirá luis:

hugo debe escribir:
hola a todos

paco debe escribir:
hola hugo

luis debe escribir:
hola hugo

hugo debe escribir:
¿alguien sabe dónde será la fiesta el sábado?

paco debe escribir:
será en la casa de donald

hugo debe escribir:
¿a qué hora?

luis debe escribir:
a las 8 PM

hugo debe escribir:
adios

paco debe escribir:
adios hugo

luis debe escribir:
adios hugo

Notar que los signos de interrogación y las letras acentuadas deberán desplegarse correctamente en la ventana de comandos de Windows.

Se **deberá** subir a la plataforma un archivo texto con el código fuente del programa desarrollado y un reporte de la tarea en formato PDF con portada, desarrollo y conclusiones como mínimo. El archivo PDF deberá incluir las capturas de pantalla de la compilación y ejecución del programa, se deberá incluir la captura de pantalla correspondiente a **cada paso** de la creación de la máquina virtual. No se admitirá la tarea si no incluye las pantallas correspondientes a cada paso del procedimiento de creación de la máquina virtual.

El nombre de la máquina virtual deberá ser el número de boleta del alumno, si el número de boleta del alumno es 12345678, entonces la máquina virtual deberá llamarse: B12345678. **No se admitirá la tarea** si la máquina virtual no se nombra como se indicó anteriormente.

Recuerden que deben **eliminar la máquina virtual** cuando no la usen, con la finalidad de ahorrar el saldo de sus cuentas de Azure.

No se admitirá la tarea si se envían archivos en formato RAR o en formato WORD.

Valor de la tarea: 20% (1.2 puntos de la segunda evaluación parcial)

- Cada alumno deberá desarrollar un sistema que calcule el producto de dos matrices cuadradas utilizando Java RMI, tal como se explicó en clase.

  Se deberá ejecutar dos casos:

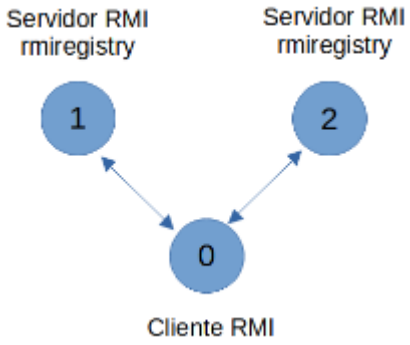
  N=8, se deberá desplegar las matrices A, B y C y el checksum de la matriz C.
  N=1000, deberá desplegar el checksum de la matriz C.

  Los elementos de las matrices A, B y C serán de tipo float y el checksum será de tipo double.

  Se deberá inicializar las matrices A y B de la siguiente manera (notar que la inicialización es diferente a la que se realizó en la tarea 3):

  A[i][j]= i + 3 * j
  B[i][j] = i - 3 * j

  El servidor RMI ejecutará en dos máquinas virtuales (nodo 1 y nodo 2) con **Ubuntu** en Azure. El programa rmiregistry ejecutará en cada nodo donde ejecute el servidor RMI. El nodo 1 calculará los productos C1 y C2 mientras que el nodo 2 calculará los productos C3 y C4.



El cliente RMI ejecutará en una tercera máquina virtual con **Ubuntu** (nodo 0). El cliente RMI inicializará las matrices A y B, obtendrá la transpuesta de la matriz B, invocará el método remoto multiplica_matrices(), calculará el checksum de la matriz C, y en su caso (N=8) desplegará las matrices A, B y C.

Se deberá utilizar las funciones que vimos en clase: separa_matriz(), multiplica_matrices() y acomoda_matriz().

Se **deberá** subir a la plataforma un archivo texto con el código fuente del programa desarrollado y un reporte de la tarea en formato PDF con portada, desarrollo y conclusiones como mínimo. El archivo PDF deberá incluir las capturas de pantalla de la compilación y ejecución del programa, se deberá incluir la captura de pantalla correspondiente a **cada paso** de la creación de las máquinas virtuales. No se admitirá la tarea si no incluye las pantallas correspondientes a cada paso del procedimiento de creación de las máquinas virtuales.

El nombre de cada máquina virtual deberá ser el número de boleta del alumno, un guión y el número de nodo, por ejemplo, si el número de boleta del alumno es 12345678, entonces el nodo 0 deberá llamarse: 12345678-0, el nodo 1 deberá llamarse 12345678-1, y así sucesivamente. **No se admitirá la tarea** si los nodos no se nombran como se indicó anteriormente.

Valor de la tarea: 30% (1.8 puntos de la segunda evaluación parcial)

- Cada alumno ejecutará el procedimiento que vimos en clase, dónde instalamos Tomcat, instalamos MySQL, y creamos un servicio web estilo REST.

  Se deberá probar el servicio web utilizando la aplicación web prueba.html tal como se explicó en clase.

  Se **deberá** subir a la plataforma el código fuente de los programas y un reporte de la tarea en formato PDF con portada, desarrollo y conclusiones como mínimo.

  El reporte PDF deberá incluir las capturas de pantalla de la compilación y ejecución del programa, se deberá incluir la captura de pantalla correspondiente a **cada paso** de la creación de la máquina virtual.

  No se admitirá la tarea si no incluye las pantallas correspondientes a cada paso del procedimiento de creación de la máquina virtual.

  El nombre de la máquina virtual deberá ser el número de boleta del alumno, si el número de boleta del alumno es 12345678, entonces la máquina virtual deberá llamarse: A12345678. **No se admitirá la tarea** si la máquina virtual no se nombra como se indicó anteriormente.

  La captura de pantallas deberá estar completa, no se admitirá la tarea si incluye imágenes que sean cortes de las capturas de pantalla.

  Recuerden que deben **eliminar la máquina virtual** cuando no la usen, con la finalidad de ahorrar el saldo de sus cuentas de Azure.

No se admitirá la tarea si se envía en formato RAR o en formato WORD.

Valor de la tarea: 20% (1.2 punto de la segunda evaluación parcial)

- Cada alumno deberá desarrollar un programa Java consola (modo carácter) cliente del servicio web que creamos en la tarea 7.

  Se deberá realizar las siguientes modificaciones al servicio web que creamos en la tarea 7:

  1. Agregar el campo id_usuario a la clase Usuario.

  2. Modificar el método web "alta_usuario", de manera que al dar de alta un usuario el método web deberá regresar al cliente el id del usuario agregado. Se deberá desplegar el id del usuario dado de alta. El campo id_usaurio es auto_increment en la base de datos, por tanto se deberá recuperar el ID inmediatamente después de ejecutar la instrucción INSERT.

  3. Modificar el método web "consulta_usuario", ahora la consulta se deberá realizar mediante el id del usuario no el email.

  4. Modificar el método web "modifica_usuario", utilizar el id del usuario en el WHERE de la instrucción UPDATE en lugar del email. No deberá ser posible modificar el id de un usuario ya que se trata de la llave primaria.

  5. Modificar el método web "borra_usuario", utilizando como clave el id del usuario no el email.

  El programa cliente deberá desplegar el siguiente menú:

  MENU

  a. Alta usuario
  b. Consulta usuario

  c. Borra usuario

  d. Salir

  Opción: _

  Las opciones deberán implementar la siguiente funcionalidad:

  - La opción "Alta usuario" leerá del teclado el email, el nombre del usuario, el apellido paterno, el apellido materno, la fecha de nacimiento, el teléfono y el género ("M" o "F"). Entonces se invocará el método web "alta_usuario". Se deberá desplegar el id del usuario dado de alta, o bien, el mensaje de error que regresa el servicio web. Notar que el método web "alta_usuario" recibe como parámetro una instancia de la clase Usuario, recordemos que esta clase se deberá definir de la siguiente manera:

```
class Usuario
{

  int id_usuario;
  String email;
  String nombre;
  String apellido_paterno;
  String apellido_materno;
  String fecha_nacimiento;
  String telefono;
  String genero;
  byte[] foto;
}
```

Para invocar el método web "alta_usuario" desde el cliente Java es necesario crear una instancia de la clase Usuario y asignar los valores a los campos (en este caso el campo "foto" será null). Una vez que se tenga el objeto de tipo Usuario se deberá utilizar **GSON** para convertir el objeto a una string JSON, entonces se deberá utilizar está string como valor del parámetro (ver más adelante cómo se enviarán los parámetros a los métodos web).

- La opción "Consulta usuario" leerá del teclado el id de un usuario previamente dado de alta. Entonces se invocará el método web "consulta_usuario". Si el usuario existe, se desplegará en pantalla el nombre del usuario, el apellido paterno, el apellido materno, la fecha de nacimiento, el teléfono y el género. Debido a que el programa no es gráfico, la foto del usuario se ignorará. Notar que el método web "consulta_usuario" regresa una string JSON la cual representa un objeto de tipo Usuario, por tanto será necesario utilizar **GSON** para convertir la string JSON a un objeto Java de tipo Usuario y posteriormente desplegar los campos del objeto (excepto el campo "foto"). Si hubo error, se desplegará el mensaje que regresa el servicio web.
- La opción "Borra usuario" leerá del teclado el id de un usuario previamente dado de alta. Entonces se invocará el método "borra_usuario" del servicio web. Se deberá desplegar "El usuario ha sido borrado" si se pudo borrar el usuario, o bien, el mensaje de error que regresa el servicio web.
- La opción "Salir" terminará el programa.

¿Cómo invocar un método web REST utilizando Java?

A continuación se muestra un ejemplo de código Java que permite invocar el método web "consulta_usuario" del servicio web **Servicio.war**

La URL que utilizaremos para acceder el método web "consulta_usuario" es la siguiente (notar que el nombre del método se escribe al final de la URL):

`http://`*`ip-de-la-máquina-virtual`*`:8080/Servicio/rest/ws/consulta_usuario`

Para que el método web pueda recibir los parámetros, cada parámetro se codifica de la siguiente manera: **nombre=valor**. Los parámetros se deben separar con un caracter **&**

Por ejemplo, si un método web tiene tres parámetros a, b y c, y los parámetros tienen los valores 10, 20 y 30, entonces se deberá enviar al método web los parámetros de la siguiente manera: "a=10&b=20&c=30", en este caso no es necesario utilizar el método URLEncoder.encode() para codificar el valor ya que se trata de un número.

El código que permite invocar el método "consulta_usuario" del servicio web Servicio.war es el siguiente:

```
URL url = new URL("http://ip-de-la-máquina-
virtual:8080/Servicio/rest/ws/consulta_usuario");

HttpURLConnection conexion = (HttpURLConnection) url.openConnection();

conexion.setDoOutput(true);

// en este caso utilizamos el método POST de HTTP
conexion.setRequestMethod("POST");

// indica que la petición estará codificada como URL
conexion.setRequestProperty("Content-Type","application/x-www-form-
urlencoded");

// el método web "consulta_usuario" recibe como parámetro el id de un
usuario, en este caso el id es 10
String parametros = "id=" + URLEncoder.encode("10","UTF-8");

OutputStream os = conexion.getOutputStream();

os.write(parametros.getBytes());

os.flush();
```

```
// se debe verificar si hubo error
if (conexion.getResponseCode() == 200)
{
  // no hubo error

  BufferedReader br = new BufferedReader(new
InputStreamReader((conexion.getInputStream())));

  String respuesta;

  // el método web regresa una string en formato JSON
  while ((respuesta = br.readLine()) != null)
System.out.println(respuesta);

}
else
{
  // hubo error
  BufferedReader br = new BufferedReader(new
InputStreamReader((conexion.getErrorStream())));

  String respuesta;

  // el método web regresa una instancia de la clase Error en formato
JSON
  while ((respuesta = br.readLine()) != null)
System.out.println(respuesta);

  // dispara una excepción para terminar el programa
  throw new RuntimeException("Codigo de error HTTP: " +
conexion.getResponseCode());
}

conexion.disconnect();
```

Notar que el código anterior se puede usar para desarrollar aplicaciones Android que requieran consumir servicios web REST.

Se deberá entregar un reporte en formato PDF que incluya las capturas de pantalla donde se muestre cada paso, desde la creación de la máquina virtual hasta la compilación y ejecución de cada opción del menú. El reporte deberá tener portada y conclusiones. Se deberá entregar también el todos los archivos utilizados incluyendo el código fuente del servicio web y el cliente.

El nombre de la máquina virtual deberá ser el número de boleta del alumno, si el número de boleta del alumno es 12345678, entonces la máquina virtual deberá llamarse: X12345678. **No se admitirá la tarea** si la máquina virtual no se nombra como se indicó anteriormente.

Las capturas de pantalla deberán ser completas, no se aceptará la tarea si incluye recortes de capturas de pantalla.

Valor de la tarea: 30% (1.8 puntos de la segunda evaluación parcial)

-
-
-
-

- **Clase del día - 12/05/2020**

  La clase de hoy vamos a iniciar con el tema "Servicios de nombres".

  Nombre, identificador y dirección

  Un **nombre**, en el contexto de un sistema distribuido, es una cadena de caracteres que hace referencia a una entidad o recurso (servidor, impresora, archivo, disco, página web,

etc).

Se entiende como **punto de acceso** a una entidad como el dispositivo desde el cual se tiene acceso a la entidad. Por ejemplo, una computadora es el punto de acceso a los archivos que contiene. Al nombre de un punto de acceso se le llama **dirección**.

Por ejemplo, la dirección de un servicio que ofrece un proceso servidor es el *end point* del servicio (dirección IP y puerto).

Un **identificador** es un nombre que tiene las siguientes propiedades:

1. El identificador hace referencia a una entidad como máximo.
2. Cada entidad tiene un identificador.
3. Un identificador siempre hace referencia a la misma entidad.

En general una dirección no es un identificador, ya que la dirección de una entidad puede cambiar (por ejemplo las direcciones IP dinámicas son modificadas por los proveedores de servicios de Internet).

Por otra parte, el nombre de dominio es en efecto un identificador.

Espacios de nombres

En general los nombres se organizan en una estructura llamada **espacio de nombres**.

Un espacio de nombres es un grafo etiquetado dirigido con dos tipos de nodos: nodos hoja y nodos directorio.

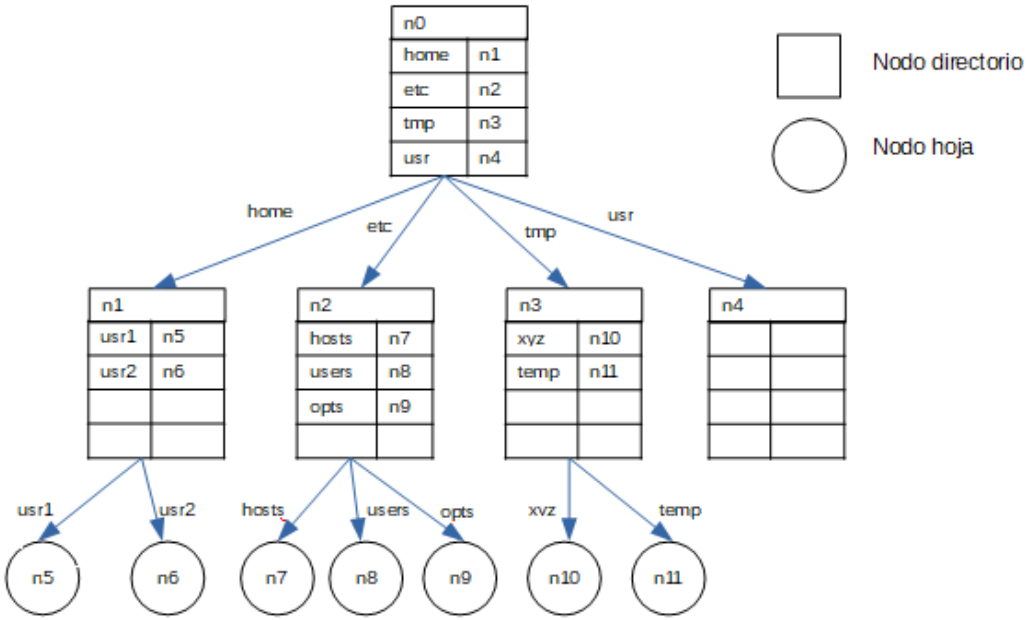
Un grafo etiquetado dirigido está compuesto por nodos conectados por arcos (o aristas). Los arcos tienen una dirección (cada arco tiene una flecha) y una etiqueta. Un arco es de **salida** si apunta afuera del nodo y es de **entrada** si apunta adentro del nodo.

Un **nodo hoja** representa una entidad con un nombre, se trata de un nodo hoja ya que no tiene arcos de salida. Este tipo de nodo guarda información sobre la entidad.

Por otra parte un **nodo directorio** contiene una tabla llamada **tabla de directorio** la cual contiene pares (etiqueta,identificador de nodo), cada etiqueta corresponde a un arco de salida del nodo directorio y cada identificador corresponde al nombre del nodo al que conecta.

Un nodo directorio puede conectar a otro nodo directorio o a un nodo hoja. Al nodo que sólo tiene arcos de salida se le llama **nodo raíz**. Un grafo que representa un espacio de nombres puede tener más de un nodo raíz, aunque por simplicidad los espacios de nombres tienen un sólo nodo raíz.

Por ejemplo, el siguiente grafo representa un sistema de archivos:



En un grafo de nombres cada nodo hoja corresponde a una entidad, en el ejemplo anterior cada entidad es un archivo.

El nombre completo de un nodo hoja en un espacio de nombres se compone de la secuencia de etiquetas de los arcos, iniciando con el nombre del nodo raíz:

N: etiqueta$_1$, etiqueta$_2$, …, etiqueta$_n$

Donde N es el primer nodo de la ruta. Por ejemplo, el archivo “xyz” tiene la siguiente ruta desde el nodo n0:

n0: etc, xyz

A esta secuencia se le llama **nombre de ruta**.

Si el primer nodo de la ruta es el nodo raíz se le llama **nombre de ruta absoluto**, de otra manera, se le llama **nombre de ruta relativo**.

En un sistema de archivos, al primer nodo se le llama generalmente disco lógico y la secuencia de etiquetas se separa por una diagonal “/”, entonces el nombre de ruta es una cadena de caracteres que corresponde a la secuencia de etiquetas.

En general, los recursos tales como procesos, dispositivos de E/S, memoria, etc. forman parte de un espacio de nombres, por tanto, también se puede aplicar cadenas de caracteres como nombres de ruta.

Árbol de nombres

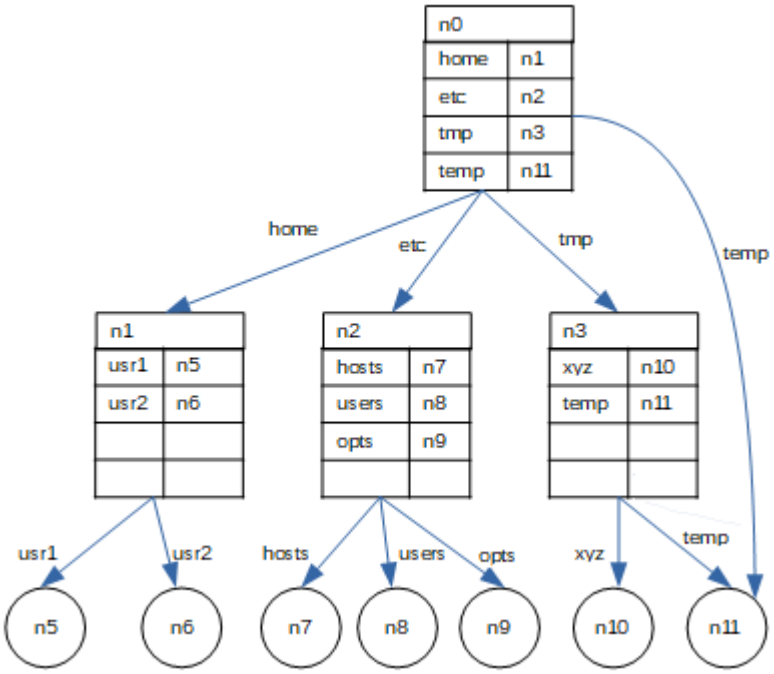
Un árbol de nombres es un grafo de nombres dónde cada nodo tiene exactamente un arco de entrada (un nodo padre), excepto el nodo raíz el cual no tiene arco de entrada.

El ejemplo mostrado anteriormente es un árbol de nombres.

Grafo dirigido no cíclico

En un grafo dirigido no cíclico un nodo puede tener más un arco de entrada, pero no se permite que haya ciclos.

En el siguiente ejemplo, podemos ver que el nodo 11 puede ser accedido utilizando dos rutas. Este es el caso de los archivos con vinculos (*links*) en los sistemas de archivos modernos. Podemos ver que no se trata de un árbol de nombres debido a que en este caso hay nodos que pueden tener más de un arco de entrada (más de un nodo padre).



Ahora vamos a ver cómo se resuelven los nombres en un espacio de nombres y cómo se implementa un espacio de nombres distribuido.

Resolución de nombres

Una de las aplicaciones de los espacios de nombres es el almacenamiento y recuperación de recursos mediante nombres.

En general, dado el nombre de ruta deberá ser posible encontrar el recurso asociado al nombre. Al proceso de búsqueda de un nombre en un espacio de nombres se le llama **resolución de nombre**.

Supongamos que tenemos un nombre de ruta de la forma:

N: etiqueta$_1$, etiqueta$_2$, etiqueta$_2$, … ,etiqueta$_n$

La resolución de nombre inicia en el nodo N, entonces se busca la etiqueta1 en la tabla de directorio del nodo N, si existe, se obtiene el identificador del nodo siguiente. Ahora se busca la etiqueta$_2$ en la tabla de directorio del nodo actual, si existe, se obtiene el identificador del nodo siguiente.

El proceso continúa hasta encontrar el nodo correspondiente a la etiqueta$_n$

Mecanismo de clausura

Se le llama **mecanismo de clausura** a la selección del nodo inicial dentro de un espacio de nombres en el cual comienza la resolución de nombre.

Un mecanismo de clausura es implícito al proceso de resolución de nombre, lo cual significa que el mecanismo de clausura debe estar definido e implementado antes de iniciar el proceso de resolución.

Por ejemplo, si el primer nodo de un nombre de ruta es el nodo raíz, entonces sabemos que el nodo raíz será un nodo directorio dónde se realizará la búsqueda de la primera etiqueta.

Si el primer nodo del nombre de ruta no es el nodo raíz, entonces deberá existir una forma de saber cómo encontrar ese nodo inicial. En un sistema de archivos de tipo Unix, si se quiere resolver utilizando un nombre de ruta relativo, el mecanismo de clausura queda definido por el directorio actual (*working directory*).

Vinculo absoluto y vinculo simbólico

Un **alias** es un segundo nombre para una misma entidad en un espacio de nombres.

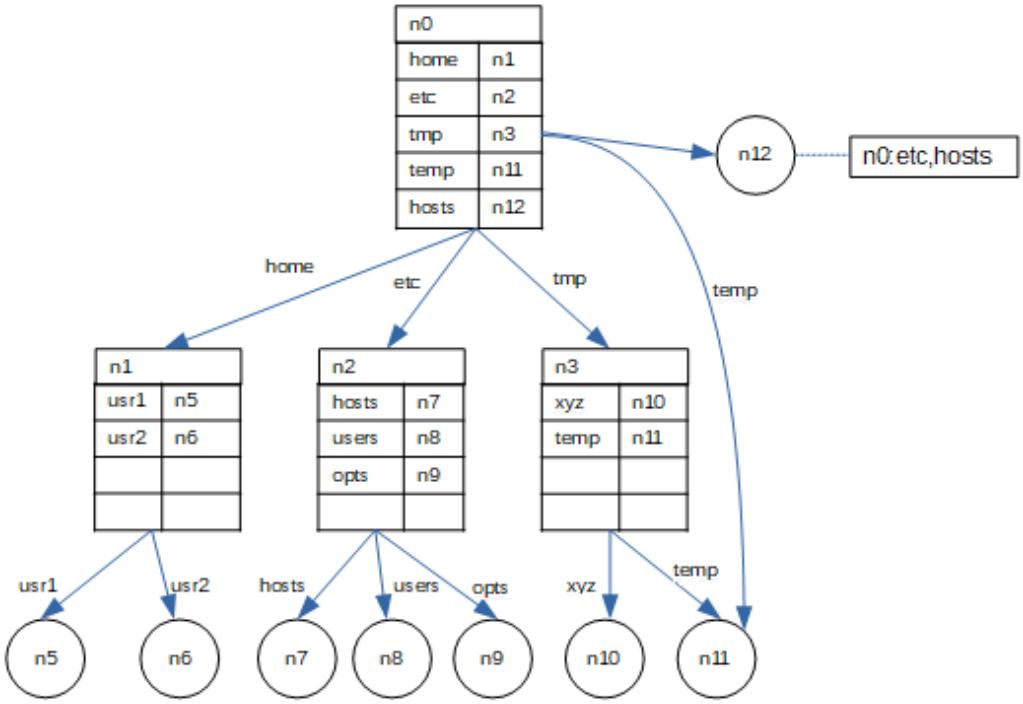
Existen dos formas de implementar un alias para una entidad: vinculo absoluto (*hard link*) y vinculo simbólico (*symbolic link*).

Un **vinculo absoluto** es simplemente un segundo nombre de ruta para una entidad en el espacio de nombres.

Un **vinculo simbólico** consiste en almacenar en un nodo hoja el nombre de ruta absoluto correspondiente a la entidad. Para encontrar la entidad se recorre el nombre de ruta hasta el nodo hoja que contiene el nombre de ruta absoluto, entonces se busca la entidad utilizando este nombre absoluto.

En el siguiente ejemplo podemos ver que el archivo “temp” puede ser resuelto mediante la ruta n0:tmp,temp y mediante la ruta n0:temp. En este caso se trata de un vinculo absoluto.

Por otra parte, el archivo “hosts” puede ser resuelto mediante la ruta n0:hosts y mediante la ruta n0:etc,hosts. En este caso se trata de un vínculo simbólico, ya que la resolución de la ruta n0:hosts lleva al nodo hoja n12 dónde se obtiene la ruta absoluta del nodo n7.



Montar un espacios de nombres

La resolución de nombres que vimos anteriormente puede ser utilizada para enlazar diferentes espacios de nombres en forma transparente.

Montar un espacio de nombres B en un espacio de nombres A consiste en hacer que un nodo directorio del espacio de nombres A incluya el identificador de un nodo directorio del espacio de nombres B.

Al nodo directorio del espacio de direcciones A que contiene el identificador del nodo externo se le llama **punto de montaje**. Así mismo, se le llama punto de montaje al nodo directorio del espacio de direcciones B.

En general, el punto de montaje externo es un nodo raíz.

El concepto de montaje de espacios de nombre permite implementar sistemas de espacios de nombres distribuidos, dónde cada computadora podría administrar un espacio de nombres local.

Para montar un espacio de nombres externo se requiere al menos de lo siguiente:

1. El nombre del protocolo de comunicación.
2. El nombre de la computadora remota.
3. El nombre del punto de montaje en la computadora remota.

El nombre del protocolo de comunicación define cómo se va a comunicar la computadora local con la computadora remota.

El nombre de la computadora puede ser la dirección IP de la computadora o bien el nombre de dominio el cual puede ser resuelto mediante un servidor de nombres de dominio (DNS).

Finalmente, deberá existir un mecanismo de clausura en la computadora remota que resuelva el punto de montaje.

Recordemos que ya utilizamos un servidor de nombres llamado rmiregistry, donde se identifican los objetos remotos mediante una URL de la forma: rmi://ip-del-servidor/nombre-del-objeto.

- **Clase del día - 13/05/2020**

  La clase de hoy veremos dos ejemplos de espacios de nombres, el sistema de archivos distribuido NFS y el sistema de nombres de dominio DNS.

  El sistema de archivos distribuidos NFS

  El sistema de archivos NFS (Network File System) permite que una computadora (cliente) tenga acceso de manera transparente a los archivos contenidos en un servidor remoto.

  Supongamos que una computadora va a acceder mediante NFS los archivos que se encuentran en el directorio /home/usuario de un servidor remoto.

  Si el dominio del servidor remoto es m4gm.com, para que el cliente tenga acceso al directorio remoto es necesario montar el espacio de nombres remoto en el espacio de nombres local.

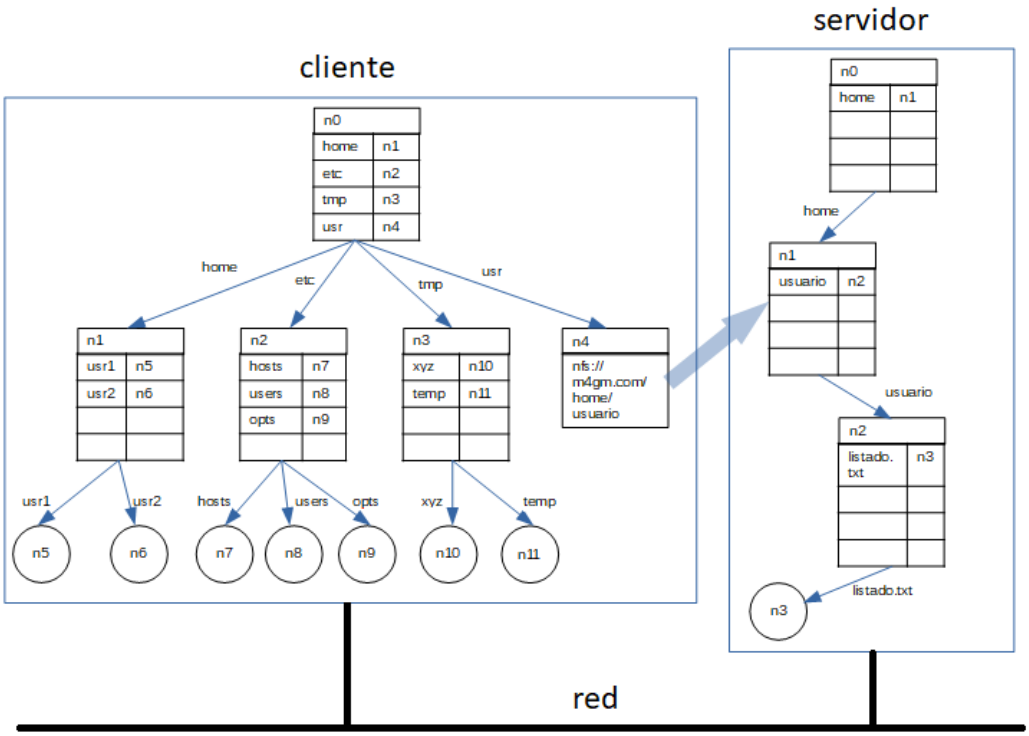
  Para montar el espacio de nombres remoto se elige un punto de montaje en el cliente, por ejemplo se puede elegir el directorio /usr. Entonces el nodo directorio correspondiente a /usr va a contener una URL que define el protocolo, el dominio del servidor y el punto de montaje en el servidor, en este caso la URL sería:

  nfs://m4gm.com/home/usuario

  En general un cliente indica qué archivo va a acceder utilizando una URL de la forma:

  nfs://dominio-o-ip-del-servidor/punto-de-montaje

  En la figura se puede ver que el nodo directorio n4 contiene la URL que define el punto de montaje del espacio de nombres remoto.



Si el cliente requiere acceder al archivo remoto /home/usuario/listado.txt solo tiene que acceder al “archivo local” /usr/listado.txt

La localización del archivo listado.txt se realiza en tres pasos:

1. El espacio de nombres local resuelve el nombre /usr
2. El dominio m4gm.com se resuelve accediendo un DNS entonces se obtiene la dirección IP del servidor.
3. El servidor resuelve el nombre /home/usuario

La ventaja de utilizar archivos remotos mediante NFS es que el usuario accede a los archivos sin preocuparse por los detalles de la comunicación entre el cliente y el servidor.

Domain Name System (DNS)

Un espacio de nombres distribuido a gran escala, como es el caso de un DNS, se organiza de manera jerárquica en tres capas.

Capa global

La capa global se compone de los nodos de más lato nivel, a saber, el nodo raíz y sus hijos. Todos los nodos en esta capa son **nodos directorio**.

Las etiquetas de los arcos son los diferentes tipos de dominio (com, org, edu, etc.). Las tablas de directorio en la capa global casi nunca se modifican.

Capa de administración

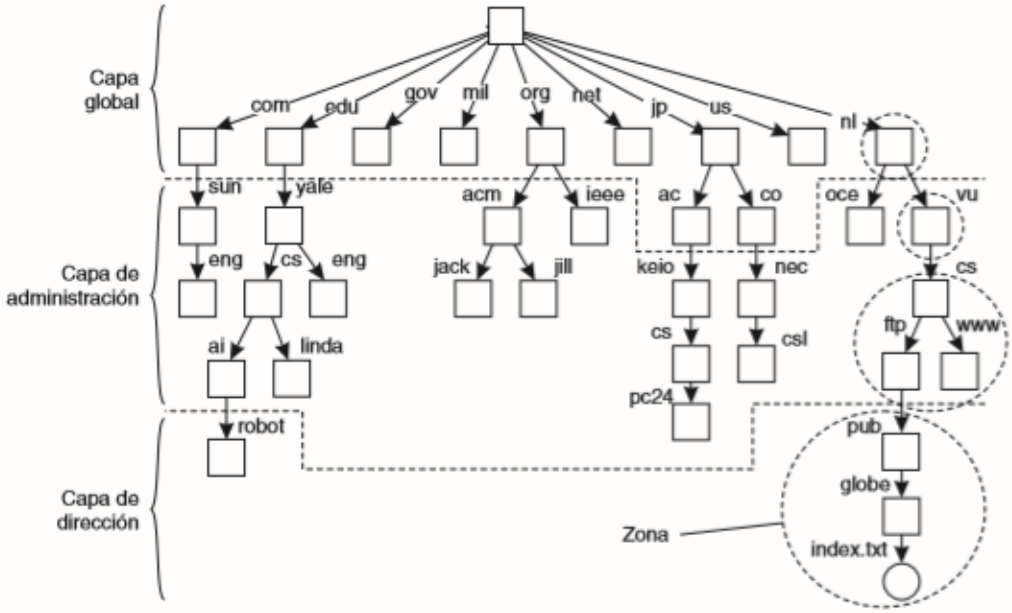
La capa de administración se compone de **nodos directorio** que son administrados dentro de una misma organización. Por ejemplo, un nodo podría corresponder al subdominio llamado “sun” y este tener un subdominio llamado “eng”, en este caso el subdominio sería eng.sun.com

Las tablas de directorio de la capa de administración se modifican poco, debido a que generalmente representan unidades administrativas dentro de una organización.

Capa de dirección

La capa de dirección se compone de nodos que pueden modificarse con cierta frecuencia. Los nodos en esta capa representan servidores con el último subdominio

La siguiente figura muestra un ejemplo del espacio de nombres de un DNS. Es esta figura se pueden ver partes del espacio de nombres llamadas zonas, las cuales se manejan mediante servidores de nombres por separado.



Ejemplo de un DNS

Fuente: Sistemas Distribuidos, Principios y Paradigmas, Andrew S. Tanenbaum, Pearson.

- **Clase del día - 17/05/2021**

La clase de hoy vamos a ver cómo realiza la resolución de nombres un DNS.

Resolución de nombre en un DNS

Cuando un usuario escribe una URL en un navegador web, se inicia un proceso de resolución de nombres para la URL, en primer lugar, se debe resolver el dominio al cual se va a conectar el navegador.

La resolución del dominio la realiza un **solucionador de nombre** dentro del sistema operativo que ejecuta el navegador.

Supongamos que el usuario escribe la siguiente URL en su navegador web:

ftp://ftp.cs.vu.nl//pub/globe/ index.html

El nombre de ruta correspondiente sería el siguiente (ver el tema Espacios de nombres):

root:nl,vu,cs,ftp,pub,globe,index.html

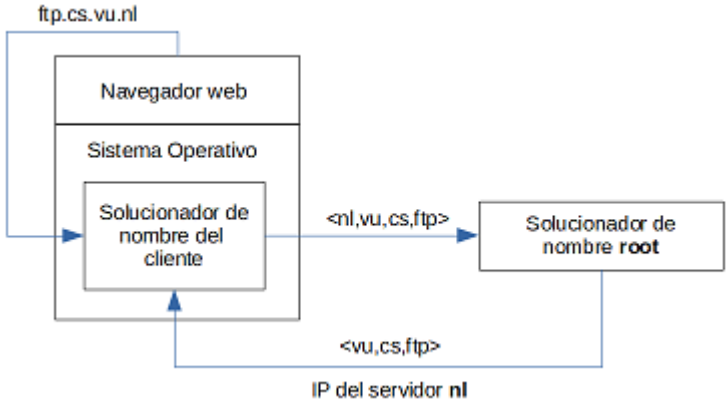
Esto significa que el usuario requiere acceder al archivo “index.html” el cual se encuentra en el directorio “/pub/globe” en el servidor cuyo dominio es “ftp.cs.vu.nl” (Ejemplo de DNS que vimos la clase anterior).

Para resolver la URL existen dos técnicas, la resolución iterativa y la resolución recursiva.

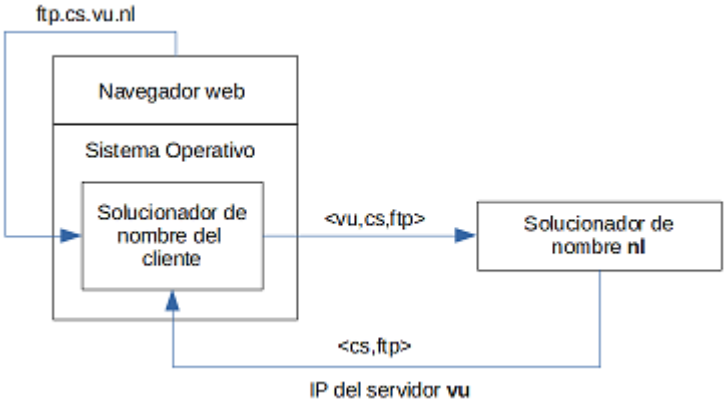
Resolución iterativa

En la resolución iterativa se ejecutan los siguientes pasos:

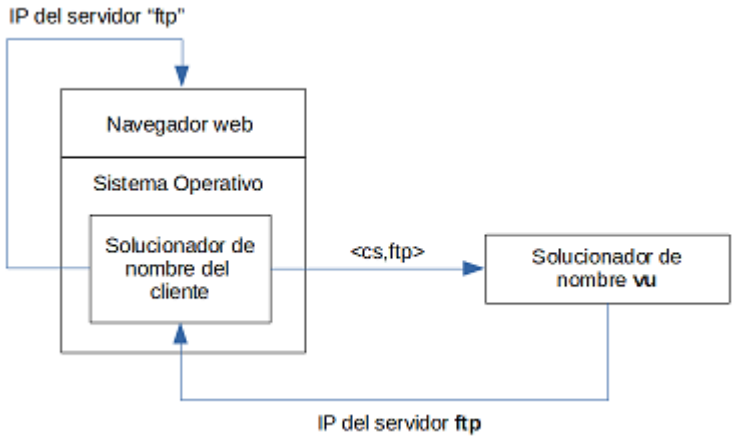
1) El solucionador de nombre del cliente se conecta a un solucionador de nombre **root** cuya dirección IP es conocida enviando la ruta <nl,vu,cs,ftp>. Este servidor resolverá el nombre hasta dónde le sea posible, en este caso solo puede resolver la etiqueta “nl”, entonces regresará al solucionador de nombre del cliente la dirección IP del solucionador de nombre **nl** y la ruta restante <vu,cs,ftp>.



2) El solucionador de nombre del cliente se conecta al solucionador de nombre **nl** enviando la ruta <vu,cs,ftp>. Este servidor solo puede resolver la etiqueta “vu”, entonces regresará al solucionador de nombre del cliente la dirección IP del solucionador de nombre **vu** y la ruta restante <cs,ftp>.



3) El solucionador de nombre del cliente se conecta al solucionador de nombre **vu** enviando la ruta <cs,ftp>. Este servidor puede resolver las etiquetas “cs” y “ftp”, entonces regresará al solucionador de nombre del cliente la dirección IP del servidor **ftp**.



4) El navegador web se conecta al servidor **ftp** enviando la ruta /pub/globe/index.html. Finalmente el servidor FTP regresa el archivo solicitado.

Resolución recursiva

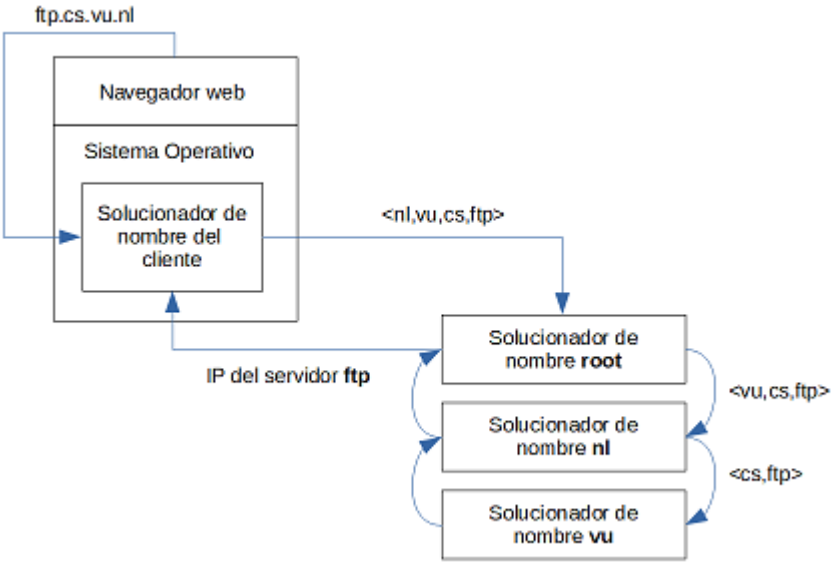
Debido a que el solucionador de nombre del cliente suele estar lejos de los solucionadores de nombres, la resolución iterativa puede ser tardara en términos de comunicación. Una alternativa a la resolución iterativa de nombres es el uso de la técnica de resolución recursiva.

En la resolución recursiva, el solucionador de nombre del cliente se comunica con el solucionador de nombre **root** enviando la ruta <nl,vu,cs,ftp>, este servidor solo puede resolver la etiqueta “nl” por tanto se comunica con el solucionador de nombre **nl** enviando el resto de la ruta <vu,cs,ftp>.

El solucionador de nombre **nl** sólo puede resolver la etiqueta “vu” por tanto se comunica con el solucionador de nombre **vu** enviando el resto de la ruta <cs,ftp>.

Finalmente, el solucionador de nombre **vu** resuelve las etiquetas “cs” y “ftp”, entonces regresará al solucionador de nombre **nl** la dirección IP del servidor **ftp**. El solucionador de nombre **nl** le envía la dirección IP al solucionador de nombre root, y este le enví la IP al solucionador de nombre del cliente.

El solucionador de nombre que resuelve la última etiqueta regresa la IP al servidor que se comunicó con él, y así sucesivamente hasta que el solucionador de nombre root regresa la IP al solucionador de nombre del cliente



La desventaja de la resolución recursiva es que representa una mayor carga en cada servidor de nombre, ya que debe mantener abierta una conexión al siguiente solucionador mientras el proceso de resolución esté en curso.

Por esta razón, los servidores de nombre de la capa global (los cuales son los que más peticiones reciben) soportan solamente la resolución iterativa.

- **Clase del día - 19/05/2021**

La clase de hoy vamos a ver cómo instalar NFS en dos máquinas virtuales en la nube.

Primeramente necesitamos crear dos máquinas virtuales (cliente y servidor) en Azure con Ubuntu 18 con las siguientes características:

- Tamaño de la memoria en la máquina virtual: 1 GB RAM
- Tipo de autenticación: Contraseña
- Tipo de disco del sistema operativo: HDD estándar

Instalación en el servidor

1. Para instalar NFS en el servidor se ejecutan los siguientes comandos:

```
sudo apt update
sudo apt install nfs-kernel-server
```

2. Crear el directorio compartido en el servidor;

```
sudo mkdir /var/nfs -p
```

3. El propietario del directorio /var/nfs es root debido a que este directorio se creó con sudo. Podemos ver el propietario del directorio /var/nfs ejecutando el comando:

```
ls -l /var
```

4. Debido a que NFS convierte el acceso del usuario root en el cliente en un acceso con el usuario "nobody:nogroup" en el servidor, es necesario cambiar el propietario y permisos del directorio creado anteriormente:

```
sudo chown nobody:nogroup /var/nfs
sudo chmod 777 /var/nfs
```

5. Podemos verificar el nuevo propietario:

```
ls -l /var
```

6. Ahora se debe registrar el directorio creado en el archivo de configuración de NFS.

6.1 Editar el archivo /etc/exports:

```
sudo vi /etc/exports
```

6.2 Agregar la siguiente línea, guardar y salir del editor (en este caso 40.76.45.28 es la IP del cliente):

```
/var/nfs 40.76.45.28(rw,sync,no_subtree_check)
```

(Para una descripción de los permisos se puede consultar Understanding the /etc/exports File)

6.3 Actualizar la tabla de file systems exportados por NFS:

```
sudo exportfs -ra
```

6.4 Para ver los file systems exportados por NFS:

```
sudo exportfs
```

6.5 Para activar la nueva configuración, se requiere reiniciar el servidor NFS:

```
sudo systemctl restart nfs-kernel-server
```

7. Ahora debemos abrir el puerto 2049 en el firewall del servidor:

Intervalos de puertos de destino: 2049
Protocolo: TCP
Nombre: puerto_nfs

Instalación en el cliente

8. Para instalar NFS en el cliente se ejecutan los siguientes comandos:

```
sudo apt update
sudo apt install nfs-common
```

9. Crear el directorio de montaje en el cliente (punto de montaje):

```
sudo mkdir -p /nfs
```

10. Montar los directorios remotos (en este caso 40.87.94.140 es la IP del servidor):

```
sudo mount -v -t nfs 40.87.94.140:/var/nfs /nfs
```

Probar el acceso a archivos remotos

11. En el servidor crear un archivo, en este caso utilizamos el editor vi:

```
vi /var/nfs/texto.txt
```

12. Escribir un texto (como el siguiente) guardar y salir de la edición:

Esta es una prueba

13. En el cliente editar el archivo /nfs/texto.txt:

```
vi /nfs/texto.txt
```

14. Agregar la siguiente linea, guardar y salir de la edición:

Esta es otra prueba

15. En el servidor desplegar el contenido del archivo /var/nfs/texto.txt:

```
more /var/nfs/texto.txt
```

16. En el cliente desplegar el contenido del directorio /nfs:

```
ls -l /nfs
```

Actividades individuales a realizar

1. Revisar la documentación del comando mount (ejecutar: man mount)
2. Revisar la documentación del comando systemctl (ejecutar: man systemctl)
3. Revisar la documentación del comando exportfs (ejecutar en el servidor: man exportfs)

- Tomando como base la clase dónde vimos cómo instalar NFS en dos máquinas virtuales en la nube, realizar lo siguiente:

    1. Crear tres máquinas virtuales con **Ubuntu 18 en la nube de Azure**.
    2. En una máquina virtual instalar un servidor NFS y en dos máquinas virtuales instalar clientes NFS.
    3. Crear en el servidor el directorio: /var/nfs
    4. Crear en cada cliente el directorio: /nfs
    5. Exportar el directorio /var/nfs a los clientes.
    6. En cada cliente montar el directorio remoto /var/nfs sobre el directorio /nfs
    7. En el cliente 1 crear un archivo de texto llamado "prueba.txt" en el directorio /nfs.
    8. Agregar al archivo "prueba.txt" el texto "esta es una prueba de NFS" y guardar el archivo.
    9. En el cliente 2 desplegar el contenido del archivo /nfs/prueba.txt utilizando el comando "more"
    10. Configurar cada cliente para que se monte automáticamente al momento del boot, el directorio /var/nfs remoto en el directorio /nfs (investigar cómo se monta un directorio remoto NFS cuando la computadora enciende; sugerencia ver: /etc/fstab).
    11. Hacer re-boot de los dos clientes
    12. En el cliente 1 desplegar el archivo /nfs/prueba.txt utilizando el comando "more"
    13. En el cliente 2 desplegar el archivo /nfs/prueba.txt utilizando el comando "more"
    14. En el cliente 2 modificar el archivo /nfs/prueba.txt, agregar al archivo el siguiente texto: "estamos agregando texto al archivo"
    15. En el cliente 1 desplegar el archivo /nfs/prueba.txt utilizando el comando "more"
    16. En el cliente 1 eliminar el archivo /nfs/prueba.txt utilizando el comando "rm"
    17. En el cliente 1 desplegar el contenido del directorio /nfs utilizando el comando "ls"
    18. En el cliente 2 desplegar el contenido del directorio /nfs utilizando el comando "ls"

Se deberá subir a la plataforma un archivo PDF que incluya portada, descripción de la tarea, la captura de las pantallas correspondientes a **cada paso** del procedimiento de creación y configuración de las máquinas virtuales así como **cada paso** de la instalación de NFS, la captura de pantallas de los pasos 3 al 18 descritos anteriormente y las conclusiones.

El nombre de cada máquina virtual deberá ser: el prefijo "NFS", el número de boleta del alumno, un guión y un número de máquina virtual, por ejemplo, si el número de boleta del alumno es 12345678, entonces la primera máquina virtual deberá llamarse: NFS12345678-0, la segunda máquina virtual deberá llamarse NFS12345678-1, y así sucesivamente. **No se admitirá la tarea** si los nodos no se nombran como se indicó anteriormente.

La captura de pantallas deberá estar completa, no se admitirá la tarea si incluye imágenes que sean cortes de las capturas de pantalla.

No se aceptará la tarea si se envía archivos en formato Word o en formato RAR.

Recuerden que deben **eliminar la máquina virtual** cuando no la usen, con la finalidad de ahorrar el saldo de sus cuentas de Azure.

Valor de la tarea: 30% (1.8 puntos de la tercera evaluación parcial)

- **Clase del día - 20/05/2021**

  En la clase de hoy vamos a comenzar con el tema de replicación.

  Los datos son el principal activo de las empresas e instituciones.

  Si bien es cierto que los sistemas informáticos son de gran importancia para automatizar los procesos en las empresas, los datos representan los objetos de negocio que mayor persistencia en el tiempo tienen, ya que ellos corresponden a lo que se sabe de los clientes, los productos, los insumos, los activos, los pasivos, las ventas, los procesos, los empleados, y muchos objetos de negocio más.

  La replicación de los datos es una estrategia utilizada para mantener copias consistentes de los datos en diferentes locaciones, con el objetivo de tener la capacidad de recuperar la operación en caso de desastre.

  ¿Por qué replicar los datos?

  Los datos se replican para satisfacer dos requerimientos no funcionales de los sistemas distribuidos: la confiabilidad y el rendimiento.

  Replicar los datos en diferentes sistemas de archivos, aumenta la **confiabilidad** del sistema, ya que si una copia de los datos falla es posible seguir trabajando con otra copia de los datos.

  Por ejemplo, en los sistemas manejadores de bases de datos se acostumbra configurar copias "espejo" de las tablas, de tal manera que cuándo se inserta, modifica o borra datos en una tabla, se realicen las mismas operaciones sobre la tabla "espejo", si la lectura de la tabla principal falla, entonces automáticamente se realiza la lectura sobre la copia "espejo", sin mayor intervención del sistema que accede a la base de datos.

  La replicación de datos también mejora el **rendimiento** de un sistema distribuido que requiere escalar en tamaño y geografía. Es una buena práctica replicar los catálogos de los sistemas (por ejemplo, los catálogos de clientes, de productos, de cuentas, etc.) ya que se trata de datos que se modifican poco, por tanto en un sistema distribuido resulta más rápido el acceso a estos datos si los tiene cerca.

  No obstante las ventajas que tiene la replicación de los datos, la principal dificultad es mantener la consistencia entre las copias.

  La **consistencia** de los datos significa que todas las copias deben tener los mismo datos.

Mantener la consistencia entre copias puede impactar el rendimiento del sistema, debido a que la actualización de las copias representa un costo en tiempo y recursos (CPU, red, almacenamiento, etc). Entonces será necesario evaluar el costo de mantener consistentes las copias y el beneficio del aumento del rendimiento que trae la replicación de los datos.

Modelos de consistencia

Un **modelo de consistencia** es un acuerdo entre los procesos que acceden un almacén de datos y el almacén de datos.

El acuerdo establece las reglas que deben obedecer los procesos cuando acceden el almacén de datos de manera que los procesos puedan tener una imagen consistente de los datos.

El **almacén de datos** puede ser una base de datos distribuida, un sistema de archivos distribuido o una combinación de ambos.

El principio en que se basa un modelo de consistencia es que si un proceso realiza una operación de lectura sobre elemento de datos, se espera leer el resultado de la última escritura sobre el mismo elemento de datos, independientemente de qué proceso realizó la escritura.

Consistencia secuencial

El modelo de consistencia secuencial fue propuesto por Lamport (1979), y dice que un almacén de datos es secuencialmente consistente si:

"*El resultado de cualquier ejecución es el mismo que si las operaciones (de lectura y escritura) de todos los procesos efectuados sobre el almacén de datos se ejecutaran en algún orden secuencial y las operaciones de cada proceso individual aparecieran en esa secuencia en el orden especificado por su programa*".

Esto significa que los procesos que ejecutan en paralelo (en un sistema distribuido) deberán "ver" la misma secuencia de operaciones de lectura y escritura al almacén de datos, y tales operaciones deberán aparecer en el mismo orden en que se ejecutan en cada proceso individual.

La consistencia secuencial corresponde a un orden parcial de las operaciones de lectura y escritura, el cual puede ser implementado mediante la relación happen-before.

Por otra parte si se requiere que las operaciones de lectura y escritura se realicen en un orden completo, sería necesario implementar un reloj global.

Consistencia de entrada

El modelo de consistencia secuencial es un modelo de granularidad fina propuesto originalmente para acceder localidades de memoria compartida en sistemas multiprocesadores, sin embargo resulta muy costoso para los sistemas distribuidos dónde los datos tienen granularidad gruesa, es decir, se accede a registros, tablas, archivos, etc.

Berchad (The Midway Distributed Shared Memory System, 1993) propuso un modelo de consistencia basado en la relación entre objetos de sincronización (locks) que protegen secciones criticas y los datos compartidos dentro de las secciones críticas.

El modelo de consistencia de entrada utiliza objetos de sincronización exclusiva y no-exclusiva (compartida) para garantizar el orden en que se ejecutan las operaciones de lectura y escritura sobre un mismo elemento de datos.

Una sección crítica comienza con una operación de adquisición del lock y termina con la liberación de lock. A estas operaciones se les llama generalmente "lock" y "unlock".

Las reglas que se debe cumplir en este modelo de consistencia son:

1. Cuando un proceso ejecuta la operación "lock" ésta debe esperar a que se realicen todas las operaciones de escritura de los datos compartidos por el proceso.

2. Un proceso no puede adquirir un lock si algún otro proceso lo adquirió ya sea en forma exclusiva o compartida.

3. Si un proceso adquirió un lock en forma exclusiva, ningún otro proceso puede adquirir el lock en forma compartida.

Estas reglas garantizan que las lecturas de datos compartidos (que se realizan dentro de una sección crítica) obtendrán los datos actualizados por la última escritura, la cual también se debió realizar dentro de una sección crítica.

Como puede observarse, no importa el orden en que se realizan las lecturas y escrituras a los elementos de datos, lo que importa es el orden en que se realizan las operaciones "lock" y "unlock".

Para lograr la consistencia de los elementos de datos compartidos, los objetos de sincronización (locks) deberán implementarse en forma global, tal como se explicó en el tema de "Sincronización y coordinación".

Veamos un ejemplo del uso de locks para sincronizar un elemento de datos compartido por dos threads que ejecutan en paralelo en una computadora con dos o más núcleos.

Supongamos que tenemos dos threads **t1** y **t2**, y cada thread ejecuta un ciclo dónde se incrementa la variable global **n**.

```
class A extends Thread
{
  static long n;
  public void run()
  {
    for (int i = 0; i < 100000; i++)
        n++;
  }
  public static void main(String[] args) throws Exception
  {
    A t1 = new A();
    A t2 = new A();
    t1.start();
    t2.start();
    t1.join();
    t2.join();
    System.out.println(n);
  }
}
```

Al ejecutar varias veces el programa anterior podemos ver que el valor final de **n** no es el mismo ¿por qué?

La instrucción **n++** copia la variable global **n** a un registro del procesador, el valor en el registro se incrementa y se escribe a la variable **n**.

Debido a que los threads **t1** y **t2** ejecutan en paralelo en una computadora con dos o más núcleos, el thread **t1** leerá y escribirá la variable **n** al mismo tiempo que el thread **t2** realiza las mismas operaciones.

Dado que la lectura que realiza **t1** no está ordenada con respecto a la escritura que hace **t2** y que la lectura que realiza **t2** no está ordenada con respecto a la escritura que hace **t1**, es posible que no se escriba algún incremento en la variable **n**, lo que produce un valor final menor a 200000.

Para resolver el problema se requiere que el programador identifique las secciones críticas y agregue las operaciones "lock" y "unlock" necesarias.

En este caso, la sección crítica es la instrucción **n++**, que es dónde se lee y escribe la variable **n** compartida por los dos threads.

Por lo tanto, es necesario ejecutar "lock" antes de **n++** y ejecutar "unlock" después.

En java se utiliza la instrucción **synchronized**(*objeto*){ *bloque-de-instrucciones* } para definir un bloque de instrucciones como sección crítica controlada por el lock que contiene el *objeto* (recordemos que en Java todos los objetos incluyen un lock).

Entonces el código del programa queda de la siguiente manera:

```
class A extends Thread
{
  static long n;
  static Object obj = new Object();
  public void run()
  {
    for (int i = 0; i < 100000; i++)
      synchronized(obj)
      {
        n++;
      }
  }
  public static void main(String[] args) throws Exception
  {
    A t1 = new A();
    A t2 = new A();
    t1.start();
    t2.start();
    t1.join();
    t2.join();
    System.out.println(n);
  }
}
```

Al ejecutar varias veces el programa anterior podemos ver que el valor final de la variable **n** siempre es 200000.

Actividades individuales a realizar

1. Compilar y ejecutar los programas que vimos.
2. ¿Qué pasa si el número de iteraciones del ciclo es 100 o 1000000 en el primer programa?

3. ¿Qué pasa si el primer programa se ejecuta en una computadora con un solo procesador? ¿Es necesario sincronizar los threads?

4. Si el segundo programa crea más threads que incrementan la variable n ¿El valor de n desplegado al final sigue siendo correcto?

Hoy vamos a jugar un kahoot en la modalidad de "challenge".

Para jugar este kahoot deberán ingresar a la siguiente URL:

Desarrollo de Sistemas Distribuidos - Espacios de nombres

Es necesario que los alumnos y alumnas accedan al kahoot con su nombre y apellido (por ejemplo JuanLopez), de manera que sea posible identificar a los ganadores de puntos extra.

Debido a que la plataforma Kahoot recientemente limitó a un máximo de 10 jugadores por kahoot, podrán jugar los primeros 10 alumnos que accedan al kahoot.

La hora límite para jugar este kahoot es 11 PM del 20 de mayo.

- **Clase del día - 24/05/2021**

En la clase anterior vimos que la replicación de los datos es una estrategia utilizada para la recuperación de un sistema en caso de desastre, así mismo, la replicación de los datos

tiene un efecto positivo en el rendimiento de un sistema, debido a que es más rápido acceder a datos cercanos.

Para mantener consistentes las diferentes copias de los datos, es necesario implementar un modelo de consistencia, el cual puede considerar diferente granularidad de los datos.

Mantener la consistencia de los datos suele ser complicado y costoso en tiempo de comunicación.

Respaldos incrementales

En el pasado, la replicación de los datos mediante respaldos era una solución medianamente buena para garantizar la continuidad de un sistema. Si un sistema fallaba, entonces se restauraba el último **respaldo** con el fin de recuperar el estado del sistema hasta un cierto punto del tiempo.

Se solía realizar **respaldos incrementales**, lo que significa que se debe diseñar una estrategia para realizar respaldos anuales, mensuales, semanales, diarios, cada hora, etc. En estas condiciones, el respaldo más frecuente solo deberá guardar los cambios realizados desde el último respaldo y no todo el almacén de datos.

No obstante, aún realizando respaldos cada hora, si el sistema falla se perderán los cambios efectuados los últimos minutos.

Para mitigar los riesgos debidos a errores humanos o el secuestro de datos (*ransomware*), los proveedores de cómputo en la nube ofrecen servicios escalables de respaldo de datos, los cuales permiten respaldar y restaurar máquinas virtuales completas o bien archivos, carpetas o bases de datos.

Respaldos continuos

Los sistemas manejadores de bases de datos (DBMS) implementan una estrategia de **respaldos continuos** de las transacciones.

A partir del inicio de una transacción (begin work), se va guardando copias de los registros que son actualizados dentro de la transacción, si el DBMS falla o la computadora se apaga, es posible recuperar la base de datos hasta un estado consistente a partir de los registros contenidos en el respaldo continuo.

El respaldo continuo de los registros modificados en una transacción también se utiliza para mantener un estado consistente de la base de datos. Si la transacción termina con **rollbak**, entonces se restablecen todos los registros modificados al estado anterior a la transacción.

Por otra parte, si la transacción termina con **commit**, entonces los cambios quedan firmes y se desecha el respaldo continuo de los registros modificados dentro de la transacción.

A la funcionalidad del DMBS que garantiza la consistencia de la base de datos se le llama **ACID** (Atomicity, Consistency, Isolation, & Durability).

En el caso de bases de datos distribuidas, el control de las transacciones distribuidas se realiza mediante un protocolo llamado two-phase commit (2PC). En este caso el respaldo continuo también se realiza en forma distribuida.

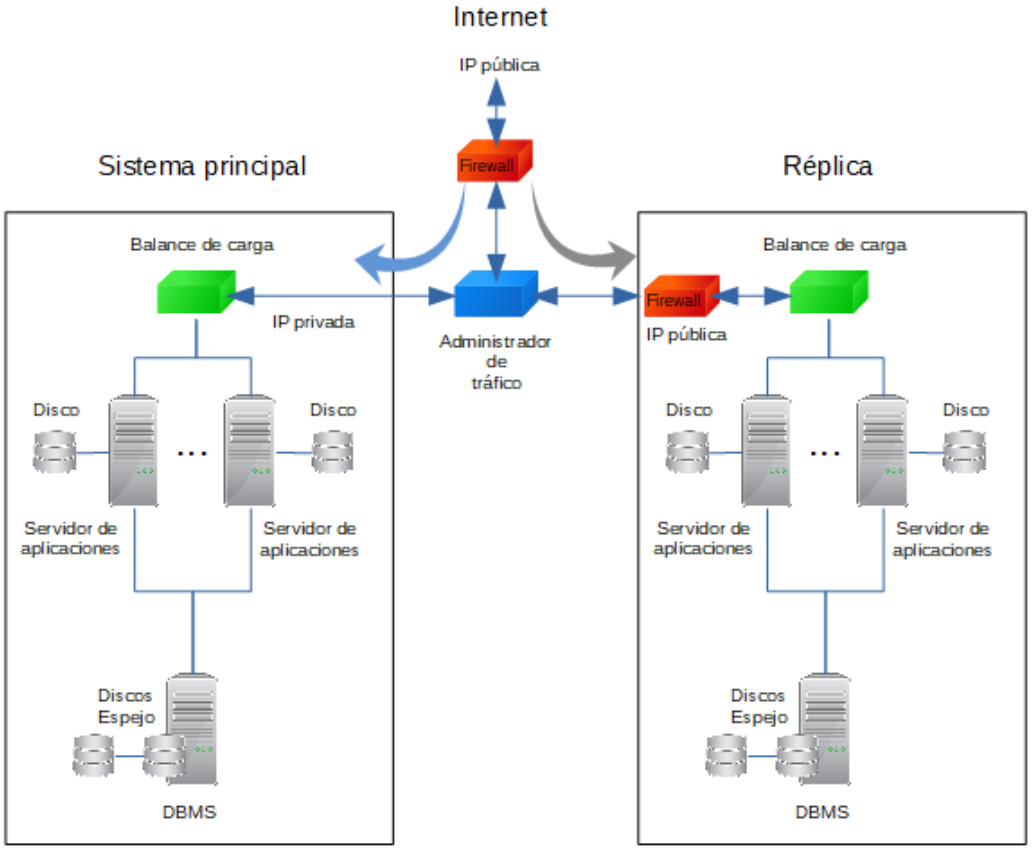
Replicación del sistema completo

En la actualidad el cómputo en la nube nos permite realizar el aprovisionamiento dinámico de recursos de forma fácil (automática), rápida y a bajo costo.

Entonces, ¿por qué no replicar el sistema completo?

Arquitectura de un sistema replicado

La siguiente figura muestra la arquitectura general de un sistema replicado:



En este caso se supone que el sistema consta de dos o más servidores de aplicaciones conectados a un servidor de bases de datos, el cual tiene implementada la replicación de datos mediante discos espejo.

El administrador de tráfico funciona como un servidor proxy transparente subrogado (*reverse transparent proxy*) el cual recibe las conexiones y peticiones de los clientes y las envía al sistema principal con copia a la réplica.

El sistema principal procesa las peticiones del cliente y envía las respuestas al administrador de tráfico, el cual las re-envía al cliente. Notar que el administrador de tráfico deberá ignorar las respuestas de la réplica.

Para garantizar la recuperación en caso de desastre catastrófico en el *site* dónde ejecuta el sistema principal, la réplica deberá ejecutar en una locación diferente, por lo tanto el administrador de tráfico y la réplica deberán tener cada uno una IP pública.

Mientras el sistema principal puede estar ejecutando en la nube o en una instalación propia de la empresa (*on-premise*), la réplilca estaría funcionando en la nube, en alguna locación geográfica diferente a la locación dónde ejecuta el sistema principal.

El administrador de tráfico puede ser un programa o un *appliance* localizado en mismo *site* dónde ejecuta el sistema principal, de manera que el administrador de tráfico se pueda conectar al sistema principal mediante una red privada.

Como puede observarse, tanto el sistema principal como la réplica realizarán las mismas transacciones sobre la base de datos (y/o el sistema de archivos), por lo que la consistencia de los datos está garantizada. Si se produce un error en la comunicación entre el administrador de tráfico y la réplica, el cliente deberá recibir un de error de comunicación.

En caso de falla del sistema principal y/o falla del administrador de tráfico, solo habrá que re-definir el dominio del sistema a la IP pública de la réplica, entonces los clientes estarán conectados directamente al sistema de respaldo. Desde luego, posteriormente será necesario: 1) sacar de producción la réplica, 2) realizar una copia espejo de la réplica, y 3) restaurar el sistema principal y el administrador de tráfico.

Azure Site Recovery

Azure Site Recovery (ASR) es un servicio que permite la replicación de máquinas virtuales entre diferentes regiones de Azure. La configuración de la replicación se lleva a cabo directamente en el portal de Azure.

ASR permite realizar la recuperación de una aplicación multi-capa (servidor web + servidor de aplicaciones + servidor de bases de datos) ejecutando en varias máquinas virtuales.

Es posible realizar pruebas de recuperación de desastres, sin impactar el ambiente de producción y a los usuarios. Así mismo, es posible mantener las aplicaciones disponibles durante el proceso de recuperación.

La clase siguiente veremos cómo replicar una máquina virtual utilizando Azure Site Recovery.

- En esta tarea vamos a realizar un ejercicio de replicación de un sistema completo, en este caso la replicación de un servidor TCP, tal como podría ser un servidor HTTP, un servidor de servicios web, un manejador de bases de datos, etc.

  Como vimos en clase, para replicar un sistema, podemos crear una máquina virtual en la nube (réplica) que procese todas las peticiones que realizan los clientes, en paralelo al proceso de las mismas peticiones que realiza el sistema principal.

  Vamos a utilizar el programa SimpleProxyServer.java el cual es un proxy simple escrito en Java, que he modificado para que funcione como un administrador de tráfico.

  Se deberá realizar lo siguiente:

  1. Crear dos máquinas virtuales en la nube de Azure con Ubuntu 18, 1 GB de RAM y disco HDD estándar.

  2. Abrir el puerto 50000 protocolo TCP en la máquina virtual 1.

  3. Abrir el puerto 50000 protocolo TCP en la máquina virtual 2.

  4. Conectar a la máquina virtual 1 (sistema principal) utilizando el programa putty.exe

  5. Instalar JDK-8 en la máquina virtual 1, ejecutando los comandos:

  sudo apt update

  sudo apt install openjdk-8-jdk-headless

  6. Utilizando el programa psftp.exe enviar a la máquina virtual 1 los archivos: Servidor2.java y SimpleProxyServer.java

  7. En la máquina virtual 1 editar el método "main" en el archivo Servidor2.java:

  ServerSocket servidor = new ServerSocket(50001);

  8. Compilar en la máquina virtual 1 los programas Servidor2.java y SimpleProxyServer.java

  9. Conectar a la máquina virtual 2 (réplica) utilizando el programa putty.exe

  10. Instalar JDK-8 en la máquina virtual 2, ejecutando los comandos:

  sudo apt update
  sudo apt install openjdk-8-jdk-headless

  11. Utilizando el programa psftp.exe enviar a la máquina virtual 2 el archivo Servidor2.java

  12. En la máquina virtual 2 editar el método "main" en el archivo Servidor2.java:

ServerSocket servidor = new ServerSocket(50000);

13. Compilar el programa Servidor2.java

14. Ejecutar el programa Servidor2.java en la máquina virtual 2:

java Servidor2&

15. Ejecutar el programa Servidor2.java en la máquina virtual 1:

java Servidor2&

16. Ejecutar el máquina virtual 1 el proxy:

java SimpleProxyServer IP-máquina-virtual-2 50000 50000 50001&

(IP-máquina-virtual-2 es la IP de la réplica, 50000 es el puerto abierto en la réplica, 50000 es el puerto abierto en el sistema principal y 50001 es el puerto en la máquina virtual 1 dónde el programa Servidor2.java recibe las peticiones. Notar que el puerto 50001 no se debe abrir en la máquina virtual 1, ya que el proxy y Servidor2.java se comunican mediante *loopback*).

17. En Windows:

17.1 Editar el programa Cliente2.java para que se conecte a la máquina virtual 1.

17.2 Compilar el programa Cliente2.java

17.3 Ejecutar el programa Cliente2.java

El cliente se conectará al programa SimpleProxyServer.java el cual a su vez se conectará al programa Servidor2.java en la máquina virtual 1 (sistema principal) y también se conectará al programa Servidor2.java que ejecuta en la máquina virtual 2 (réplica).

El programa Servidor2.java que ejecuta en la máquina virtual 1 enviará una respuesta al programa SimpleProxyServer.java y este a su vez enviará la respuesta al cliente.

Se deberá subir a la plataforma un **archivo PDF** que incluya portada, la captura de las pantallas correspondientes a cada paso del procedimiento y conclusiones.

Se **deberá** subir a la plataforma el código fuente de los programas utilizados en esta tarea y un reporte de la tarea en formato PDF incluyendo portada, desarrollo y conclusiones como mínimo.

El archivo PDF deberá incluir la captura de pantalla correspondiente a **cada paso** de la creación de las máquinas virtuales, incluyendo la configuración de los discos y la red. **No se admitirá la tarea** si no incluye la captura de las pantallas correspondientes a cada paso del procedimiento de creación y configuración de las máquinas virtuales.

El nombre de cada máquina virtual deberá incluir el número de boleta del alumno, un guión y un número de nodo, por ejemplo, si el número de boleta del alumno es 12345678, entonces la primera máquina virtual deberá llamarse: R12345678-0, y la segunda máquina virtual deberá llamarse R12345678-1. **No se admitirá la tarea** si las máquinas virtuales no se nombran como se indicó anteriormente.

Valor de la tarea: 30% (1.5 puntos de la tercera evaluación parcial)

- 5. Cómputo en la nube

5. Cómputo en la nube

- **Clase del día - 26/05/2021**

En el pasado el aprovisionamiento de recursos informáticos *On-premise* (en las instalaciones de la empresa) representaba el concurso de diferentes proveedores de bienes y servicios, como eran los representantes de ventas, ingenieros de pre-venta, fabricantes de los equipos, fabricante del sistema operativo, fabricante de la base de datos, agentes aduanales, transportistas, instaladores del *site*, proveedor de energía, proveedor de comunicaciones, instaladores del hardware, instaladores del software, entre otros.

Entonces, el aprovisionamiento de recursos informáticos era un proceso complejo y tardado, el cual culminaba con el sistema en producción.

Después había que re-aprovisionar cuándo crecían las necesidades de la empresa.

Cómputo en la nube

En 2006 aparece en la revista Wired el artículo The Information Factories de George Gilder que describe un nuevo modelo de arquitectura basado en una infraestructura de cómputo ofrecida como servicios virtuales a nivel masivo, a este nuevo modelo se le llamó *cloud computing* (cómputo en la nube).

El concepto clave en el cómputo en la nube es el "servicio", así, se ofrece infraestructura virtual y física como servicio (IaaS: Infrastructure as a Service), DBMS, plataformas de desarrollo y pruebas como servicio (PaaS: Platform as a Service), aplicaciones de software como servicio (SaaS: Software as a Service) y otros servicios con la terminación "as a Service", como Data as a Service (DaaS), Disaster Recovery as a Service (DRaaS), entre otros.

La elasticidad en la nube

Debido a que el cómputo en la nube está basado fundamentalmente en la virtualización de los recursos informáticos, este modelo de arquitectura ofrece una ventaja única, la posibilidad de hacer crecer y decrecer los recursos aprovisionados.

Supongamos un servicio de streaming bajo demanda, como es el caso de Netflix. En este tipo de servicio la demanda crece los fines de semana y decrece los días entre semana. Si el proveedor del servicio no aprovisiona los recursos suficientes para atender la demanda del fin de semana, entonces muchos usuarios se quedarán sin servicio.

Por otra parte, si el proveedor del servicio aprovisiona los recursos necesarios para atender a sus usuarios el fin de semana, estos recursos estarán sub-utilizados los días entre semana, lo cual resulta en pérdidas económicas.

Sin lugar a dudas, el éxito que han alcanzado las empresas proveedoras de streaming bajo demanda, se debe a que su modelo de negocio está basado en la posibilidad que les ofrece la nube para crecer y decrecer los recursos aprovisionados, a esta característica de la nube se le llama *elasticidad*.

El cómputo elástico es la habilidad de hacer crecer y decrecer rápidamente la capacidad de cómputo (CPUs), la memoria y el almacenamiento para adaptarse a la demanda.

Para implementar el cómputo elástico se utilizan herramientas de monitoreo, las cuales aprovisionan y des-aprovisionan recursos conforme son necesarios, sin detener la operación.

Nube pública, nube privada y nube híbrida

Se dice que la nube pública es un conjunto de servicios de TI ofrecidos mediante recursos (servidores, almacenamiento, red) propiedad de un proveedor de servicios en la nube, como es el caso de AWS, Azure, Softlayer, Oracle, Google, etc.

Por otra parte, se entiende como nube privada aquellos servicios ofrecidos a partir de la virtualización de los recursos propiedad de la misma empresa.

En este mismo orden de ideas, la nube híbrida sería la mezcla de servicios de nube pública y servicios de nube privada.

Ver: ¿Qué es la nube pública, privada e híbrida?

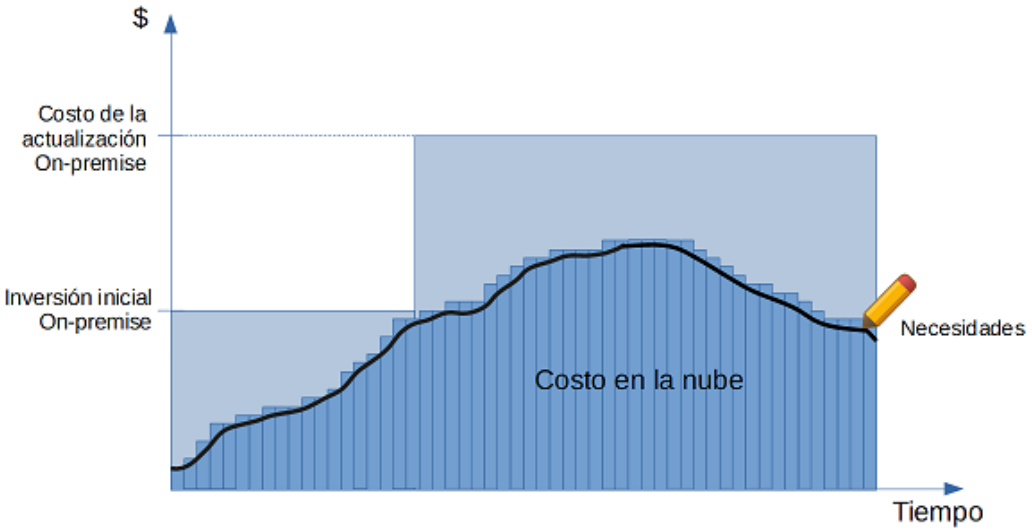
Sin embargo, hay proveedores de nube que afirman que no existe tal cosa como "nube privada", ya que el tema de elasticidad se ve acotado en un equipo "privado" debido a la limitada escalabilidad.

En cambio, en la nube pública la escalabilidad es casi ilimitada, ya que si se agotan los recursos de un data center, sin ningún problema se puede escalar a otro data center, ya sea del mismo proveedor o de otros proveedores.

Una característica importante de la nube (pública) es "pagar por lo que se usa", esto significa que solo se paga por los recursos aprovisionados.

En el caso de un centro de cómputo tradicional (también es el caso de la "nube privada") la empresa paga por todo el equipo, lo use a toda su capacidad o no.

Consideremos la siguiente gráfica:



Escenario On-premise

Al inicio los requerimientos informáticos de la empresa son pocos, sin embargo ésta debe hacer una inversión inicial grande ya que debe adquirir el equipo que le permita operar durante un periodo de tiempo determinado. Para saber cuál será la inversión inicial se deberá hacer una *planeación de capacidad*.

Suponiendo que la empresa crece, entonces en determinado momento deberá realizar la actualización de su equipo informático. Entonces se deberá hacer una planeación de capacidad para establecer el tamaño de la inversión, estimando un tiempo determinado de operación.

No obstante las previsiones, la empresa podría decrecer, por tanto los requerimientos informáticos también decrecen. El costo de la actualización no se recupera, incluso, en poco tiempo el equipo se hará obsoleto.

Escenario nube

Ahora supongamos que la empresa utiliza servicios en la nube. Entonces la inversión inicial es mínima, ya que la empresa contratará los servicios indispensables para operar. Conforme la empresa crece, la elasticidad de la nube permite adecuar el tamaño de la infraestructura informática de acuerdo a las necesidades.

Si la empresa decrece, también decrecen sus necesidades y el costo de la infraestructura informática. En la gráfica, las barras corresponden al costo de los servicios de nube. Podemos ver que este costo se ajusta a las necesidades de la empresa.

Podemos concluir que al utilizar los servicios de nube se optimiza la relación costo/beneficio, ya que solo pagamos lo que realmente usamos.

Veamos algunos ejemplos de elasticidad en Azure.

Nota importante. Es recomendable detener la máquina virtual antes de cambiar la configuración ya que es posible que se experimenten problemas de conexión mediante SSH y problemas de conexión a los servicios que ofrece la máquina virtual (HTTP, HTTPS, etc.). Si es el caso, el problema de conexión puede resolverse deteniendo la máquina virtual y volviendo a encenderla para cambiar la IP pública. Así mismo, se puede re-establecer la contraseña del usuario seleccionando la opción "Restablecer contraseña" en la sección "Soporte y solución de problemas".

Cambio del tamaño de una máquina virtual

Para cambiar el número CPUs virtuales y/o el tamaño de la memoria RAM de una máquina virtual:

1. Ejecutar el portal de Azure.
2. Seleccionar la opción "Maquinas virtuales".
3. Seleccionar la máquina virtual a modificar.
4. En el menú "Configuración" seleccionar la opción "Tamaño".
5. Seleccionar el tamaño requerido (vCPU, RAM)
6. Dar click al botón "Cambiar tamaño"
7. Nota: si la máquina virtual está en ejecución podría reiniciarse al cambiar el tamaño
8. Dar click en la ca**m**pana de notificaciones para verificar que el cambio se haya realizado con éxito.

Cambio del tamaño del disco de S.O.

Para cambiar el tamaño del disco de sistema operativo de una máquina virtual:

1. Seleccionar la máquina virtual.
2. Detener la máquina virtual.
3. Seleccionar "Discos".
4. Seleccionar el disco a cambiar de tamaño.
5. Seleccionar "tamaño y rendimiento" en el menú Configuración.
6. Seleccionar el nuevo tamaño del disco.
7. Dar clic en el botón "Cambiar tamaño".
8. Para encender la máquina virtual seleccionar la máquina virtual en el menú Inicio -> Máquinas virtuales y seleccionar Iniciar.

Actividades individuales a realizar

Vamos a jugar un kahoot en la modalidad de "challenge".

Para jugar este kahoot deberán ingresar a la siguiente URL:

Desarrollo de Sistemas Distribuidos - NFS y DNS

Es necesario que los alumnos y alumnas accedan al kahoot con su nombre y apellido (por ejemplo JuanLopez), de manera que sea posible identificar a los ganadores de puntos extra.

Debido a que la plataforma Kahoot recientemente limitó a un máximo de 10 jugadores por kahoot, podrán jugar los primeros 10 alumnos que accedan al kahoot.

La hora límite para jugar este kahoot es 11 PM del 26 de mayo.

- **Clase del día - 27/05/2021**

En la clase de hoy vamos a ver el servicio de respaldos en la nube de Azure.

Microsoft Azure Backup

Azure Backup es un servicio de respaldos en la nube que no requiere que el usuario cuente con el personal para administrar los respaldos ni la instalación de una infraestructura de almacenamiento.

Azure Backup permite respaldar y restaurar archivos, directorios, máquinas virtuales completas, bases de datos de SQL Server, o archivos y directorios On-Premise (mediante un agente que ejecuta en el equipo local).

Para proteger al usuario de ataques de *ransomware* (secuestro de datos) Azure Backup implementa la autenticación multi-factor. Así mismo, genera alertas si se detecta actividad sospechosa de respaldo o restauración de los datos del usuario.

Cuando los respaldos son borrados, Azure Backup retiene una copia de los datos por dos semanas, de manera que el usuario puede recuperar los respaldos en caso de borrado accidental.

Una característica importante de Azure Backup es que no limita la transferencia de datos de entrada o de salida. La transferencia de datos de salida se refiere a los datos que son transferidos desde el almacén (*vault*) de Recovery Services cuando se realiza una recuperación.

Un almacén de Recovery Services es un contenedor lógico que almacena los datos del recurso a proteger, por ejemplo, una máquina virtual.

Cada vez que se ejecuta el proceso de respaldo (*backup job*) para un recurso (p.e. una máquina virtual), se crea un punto de restauración dentro del almacén de Recovery Services, entonces es posible utilizar cualquier punto de restauración para recuperar los datos en un punto dado del tiempo.

Por omisión, el almacen de Recovery Services se crea como *Geo-Redundant storage*, lo cual asegura que los datos respaldados se replicarán en una región a cientos de kilómetros de la región actual.

Es posible crear políticas de respaldo (también llamadas directivas de copía de seguridad), dónde se defina cuándo se ejecuta el proceso de respaldo (*backup job*) y por cuánto tiempo se almacenarán los puntos de restauración.

Por omisión, la política de respaldo *DailyPolicy* ejecuta un respaldo diario y mantiene los puntos de restauración por 30 días.

Azure Backup tiene costo (

Azure Backup pricing

) sin embargo es más barato que aprovisionar una máquina virtual con discos de almacenamiento para ejecutar los respaldos y mantener los datos, además de otras ventajas como la replicación automáticamente en otra región geográficamente alejada, los mecanismos de seguridad adicionales como las notificaciones de actividad sospechosa y la autenticación multi-factor, la creación de puntos de restauración de la máquina virtual en diferentes tiempos sin tener que sacarla de producción, entre otras ventajas.

Habilitar el respaldo de una máquina virtual en Azure

1. Seleccionar la máquina virtual en el portal de Azure.

2. Seleccionar la opción "Backup" en el menú de operaciones.

3. Crear un almacén de Recovery Services.

4. Seleccionar el grupo de recursos dónde se colocará el almacen.

5. Seleccionar la política de respaldo, por omisión DailyPolicy, o dar click en "Crear una nueva directiva" para crear una nueva política de respaldo.

Si se crea una nueva política se puede definir la frecuencia de respaldo (diario o semanal), la hora en la que se realizará el respaldo y el tiempo que se conservará los puntos de restauración.

6. Dar click en el botón "Habilitar Backup"

7. Dar click en la campana de notificaciones para verificar que se haya realizado la implementación del proceso de respaldo.

Iniciar un respaldo completo

Anteriormente vimos cómo habilitar un proceso de respaldo (*backup job*) para realizar un respaldo diario de una máquina virtual completa a cierta hora del día.

Para iniciar al momento el respaldo completo de la máquina virtual:

1. Seleccionar la máquina virtual en el portal de Azure. Seleccionar "Backup" en el menú de operaciones.

2. Seleccionar "Realizar copia de seguridad ahora" para crear el primer respaldo completo de la máquina virtual. Los subsecuentes respaldos automáticos serán incrementales.

3. Indicar la fecha de retención de la copia de seguridad o aceptar la fecha establecida en la política de respaldo utilizada (por omisión, 30 días).

4. Dar click en el botón "Aceptar"

5. Dar click en la campana de notificaciones para verificar que se haya iniciado el respaldo de la máquina virtual.

6. Para ver el progreso del respaldo seleccionar la opción "Ver todos los trabajos" en la página "Backup" de la máquina virtual. Seleccionar la opción "Actualizar" para refrescar la pantalla que muestra el estado del proceso de respaldo. El respaldo ha terminado cuando se despliega "Completada".

El tiempo que tarda el respaldo depende del número de procesadores virtuales, el tamaño de la memoria RAM y el tamaño del disco en la máquina virtual.

Una vez terminado el respaldo en la página "Backup" de la máquina virtual aparecerá el punto de restauración creado.

Restaurar una máquina virtual

Para restaurar una máquina virtual completa:

1. Seleccionar la máquina virtual en el portal de Azure.

2. Seleccionar la opción "Backup" en el menú de "Operaciones".

3. Seleccionar la opción "Restaurar VM".

4. En "Punto de restauración" dar click en la opción "Seleccionar".

5. Seleccionar el punto de restauración y dar click en el botón "Aceptar".

6. En "Tipo de restauración" seleccionar "Crear una nueva máquina virtual".

7. Ingresar el nombre de la nueva máquina virtual.

8. Seleccionar la red virtual.

9. Seleccionar la ubicación del almacenamiento provisional. Esta cuenta de almacenamiento se utilizará temporalmente durante la restauración. Si no se encuentra una cuenta de almacenamiento será necesario crearla de la siguiente manera (cerrar la ventana actual "Restauración de la máquina virtual"):

9.1 En la ventana de búsqueda de Azure escribir: cuentas de almacenamiento
9.2 Dar clic en la opción +Agregar.
9.3 Seleccionar el grupo de recursos de la máquina virtual.
9.4 Ingresar un nombre para la cuenta de almacenamiento (no debe existir en Azure).
9.5 Seleccionar la misma ubicación del *vault* (almacén de Recovery Services) en el procedimiento **Habilitar el respaldo de una máquina virtual en Azure**.
9.6 En "Replicación" seleccionar "Almacenamiento con redundancia local (LRS)"

9.7 Dar click en el botón "Revisar y crear".
9.8 Dar click en el botón "Crear".

9.9 Ir al primer paso del procedimiento **Restaurar una máquina virtual**.

10. Dar click en el botón "Restaurar".

11. Dar click en la campana de notificaciones para verificar que se haya iniciado la restauración de la máquina virtual.

12. Para ver el progreso de la restauración seleccionar la opción "Ver todos los trabajos" en la página "Backup" de la máquina virtual. Seleccionar la opción "Actualizar" para refrescar la pantalla que muestra el estado del proceso de restauración.

Una vez terminada la restauración de la máquina virtual, la nueva máquina virtual aparecerá en la lista de máquinas virtuales en el portal de Azure.

Para conectar con la nueva máquina virtual se utilizará las mismas credenciales (usuario y contraseña) definidas para la máquina virtual respaldada.

Podemos verificar que la configuración de la nueva máquina virtual es idéntica a la configuración de la máquina virtual respaldada.

Eliminar un proceso de respaldo

Para eliminar un proceso de respaldo y los puntos de respaldo asociados:

1. Seleccionar la máquina virtual en el portal de Azure.

2. Seleccionar la opción "Backup" en el menú de "Operaciones".

3. Seleccionar "Detener copia de seguridad". Si no se ve la opción presionar los tres puntos ...

4. Seleccionar la opción "Retener datos de copia de seguridad" o bien "Eliminar datos de copia de seguridad".

5. Ingresar el nombre del elemento de copia de seguridad, es este caso el nombre de la máquina virtual respaldada.

6. Opcionalmente se puede indicar el motivo por el cual se va a eliminar el proceso de respaldo. También es posible escribir algún comentario en la ventana "Comentarios".

7. Dar click en el botón "Detener copia de seguridad".

8. Dar click en la campana de notificaciones para verificar que se haya detenido el proceso de copia de seguridad y en su caso, se haya eliminado los datos.

9. Para eliminar el almacén de Recovery Services (*vault*) hacer lo siguiente:

Nota importante: Para eliminar el almacén de Recovery Services es necesario que hayan pasado 14 días desde el último respaldo, ya que la retención de los datos es por dos semanas: "*Recovery Services vault cannot be deleted as there are backup items in soft deleted state in the vault. The soft deleted items are permanently deleted after 14 days of delete operation*". Fuente: Portal de Microsoft Azure.

9.1 Ir al inicio del portal de Azure.

9.2 Seleccionar "Todos los recursos.

9.3 Seleccionar el *vault* (almacén de Recovery Services) a eliminar (tener la precaución de seleccionar el almacén correcto).

9.4 Seleccionar también la cuenta de almacenamiento, si se creó la cuenta de almacenamiento en el paso 9 del procedimiento **Restaurar una máquina virtual**.

9.5 Seleccionar la opción "Eliminar".

9.6 Confirmar la eliminación del almacén dando click al botón "Sí"

9.7 Dar click en la campana de notificaciones para verificar que se haya eliminado el almacén.

Actividades individuales a realizar

Ver los videos:

Ver las páginas:

How to restore Azure VM data in Azure portal

- Cada alumno creará una máquina virtual en la nube de Azure y realizará los siguientes procedimientos que vimos en clase:

    1. Habilitar el respaldo de la máquina virtual.
    2. Iniciar un respaldo completo.
    3. Restaurar la máquina virtual.
    4. Eliminar el proceso de respaldo.

Se deberá entregar un reporte en **formato PDF** que incluya la captura de pantalla correspondiente a **cada paso** de los procedimientos listados anteriormente. El reporte deberá incluir además portada y conclusiones.

Nota 1. Debido a que la eliminación del almacén de Recovery Services (*Recovery Services vault*) tarda 14 días después del último respaldo, se deberá incluir la captura de pantalla del mensaje que envía Azure cuando se trata de eliminar el almacén.

El nombre de la máquina virtual deberá ser: el prefijo "BAK" y el número de boleta del alumno, si el número de boleta del alumno es 12345678, entonces la máquina virtual deberá llamarse: BAK12345678. **No se admitirá la tarea** si la máquina virtual no se nombra como se indicó anteriormente.

Nota 2. El nombre de la máquina virtual a crear en la restauración puede ser BAK12345678-2, si el número de boleta del alumno es 12345678.

Las capturas de pantallas deberán estar completas, no se admitirá la tarea si incluye imágenes que sean cortes de las capturas de pantalla.

No se admitirán tareas en formato RAR o en formato Word.

Valor de la tarea: 20% (1.2 puntos de la tercera evaluación parcial).

- **Clase del día - 31/05/2021**

La clase de hoy vamos a ver cómo crear la imagen de una máquina virtual en Azure y cómo crear máquinas virtuales a partir de la imagen.

Notas importantes

1. La captura de la imagen de una máquina virtual inutiliza la máquina virtual ya que una máquina virtual generalizada no se puede iniciar o modificar.

2. El des-aprovisionamiento (generalización) de una máquina virtual no implica que se borre toda la información confidencial que pudiera existir en la máquina virtual. Es muy importante considerar lo anterior si se va a re-distribuir la imagen de la máquina virtual.

3. El des-aprovisionamiento de una máquina virtual no elimina el archivo /etc/resolv.conf

4. El des-aprovisionamiento de una máquina virtual deshabilita la contraseña de root.

5. La opción +user del comando waagent elimina la última cuenta creada en la máquina virtual incluyendo el directorio del usuario. Si se desea conservar el usuario y el directorio,

no se deberá utilizar la opción +user al des-aprovisionar la máquina virtual mediante el comando waagent.

Crear la imagen de una máquina virtual

Para des-aprovisionar (generalizar) la máquina virtual utilizaremos el agente **waagent** el cual elimina los datos específicos de la máquina virtual.

1. Ejecutar el programa putty.exe

2. En el campo "Host Name (or Ip address)" ingrear la IP pública de la máquina virtual, dar click al botón "Open" y dar click al botón "Sí" en la ventana PuTTY Security Alert.

3. Ingresar el login del usuario (por ejemplo ubuntu) y el password.

4. Para des-aprovisionar la máquina virtual y eliminar la última cuenta de usuario creada incluyendo el directorio del usuario, ejecutar el comando:

sudo waagent -deprovision+user

Si se quiere conservar en la imagen la última cuenta de usuario creada, ejecutar el comando:

sudo waagent -deprovision

5. En el portal de Azure seleccionar la máquina virtual que se quiera capturar como imagen.

6. Seleccionar la opción "Captura".

7. Marcar la casilla "Eliminar automáticamente esta máquina virtual después de crear la imagen", ya que una máquina virtual generalizada no se puede iniciar o modificar.

8. Ingresar el nombre de la máquina virtual a capturar.

9. Dar click en el botón "Crear".

10. Dar click en la campana de notificaciones para verificar que se haya creado la imagen de la máquina virtual.

Crear una máquina virtual a partir de una imagen

1. En el portal de Azure seleccionar la imagen de la máquina virtual.

2. Seleccionar la opción "+Crear máquina virtual".

3. Seleccionar el grupo de recursos dónde se creará la máquina virtual.

4. Ingresar el nombre de la máquina virtual.

5. Seleccionar el tamaño de la máquina virtual.

6. Seleccionar el tipo de autenticación (Clave pública SSH o Contraseña). En su caso, ingresar el usuario y contraseña.

7. Dar click en el botón "Siguiente: Discos >"

8. Seleccionar el tipo de disco del sistema operativo (p.e. HDD estándar).

9. Si no hay otra configuración que se quiera realizar, dar click en el botón "Revisar y crear".

10. Dar click en el botón "Crear".

Actividades individuales a realizar

Ver las páginas:

Creación de una imagen administrada de una máquina virtual o un disco duro virtual

Información y uso del agente de Linux de Azure

Ubuntu manpages resolvconf

En la clase de hoy vamos a jugar un kahoot en la modalidad de "challenge".

Debido a que la plataforma Kahoot recientemente limitó a un máximo de 10 jugadores por kahoot, podrán jugar los primeros 10 alumnos que accedan al kahoot.

El kahoot se habilitará a la hora de clase, de acuerdo al horario presencial.

- Para jugar el kahoot deberán ingresar al siguiente enlace:

  Es necesario que los alumnos y alumnas ingresen su "nickname" como su nombre y apellido (por ejemplo JuanLopez), de manera que sea posible identificar a los ganadores de puntos extra.

  La hora límite para jugar este kahoot es 11:00 PM del 31 de mayo.

- **Clase del día - 02/06/2021**

  La clase de hoy vamos a ver el tema de replicación en la nube mediante Azure Site Recovery.

  Azure Site Recovery

  Azure Site Recovery (ASR) es un servicio que permite la replicación de máquinas virtuales a diferentes regiones de Azure. La configuración de la replicación se lleva a cabo directamente en el portal de Azure.

  ASR permite realizar la recuperación de una aplicación multi-capa (servidor de aplicaciones+base de datos+...) ejecutando en varias máquinas virtuales.

  Es posible realizar pruebas de recuperación de desastres, sin impactar el ambiente de producción y a los usuarios. Así mismo, es posible mantener las aplicaciones disponibles durante el proceso de recuperación.

  Crear la máquina virtual

  Crear un máquina virtual RedHat Linux 8.1 con 2 GB de RAM (B1ms) disco HDD estándar.

  En este caso la máquina virtual va tener RedHat Linux 8.1 debidoa a que la replicación no está soportada para todas las versiones de los sistemas operativos. Ver la sección "Azure VM Requeriments" en

  Support matrix for Azure VM disaster recovery between Azure regions

  .

  Replicar una máquina virtual

  Después de crear la máquina virtual presionar el botón "Ir al recurso".

  Seleccionar la opción "Recuperación ante desastres" en el menú que queda a la izquierda de la pantalla.

  Seleccionar la región de destino si se quiere cambiar la región recomendada.

  Dar click en el botón "Revisar e iniciar replicación".

  Se muestra entonces la configuración del origen y el destino. Se puede ver que se creó un nuevo grupo de recursos (prueba-asr) y una nueva red virtual (prueba-vnet-asr).

Se creó un Disco administrado de réplica (en este caso un disco HD estándar), el cual es réplica del disco administrado de origen (en este caso HD estándar).

Dar click al botón "Iniciar replicación".

Revisar las Notificaciones (dar click en la campana que se encuentra en la barra superior) para verificar que se haya habilitado la replicación correctamente.

Si hay un error por falta de espacio en el disco de sistema operativo (S.O.), se puede aumentar el tamaño del disco. Este tema lo vimos en una clase previa**: Cambio del tamaño del disco de S.O.**

Actividades individuales a realizar

1. Revisar el artículo: Azure Site Recovery

2. Revisar el artículo: Inicio rápido: Configuración de la recuperación ante desastres en una región secundaria de Azure de una máquina virtual de Azure

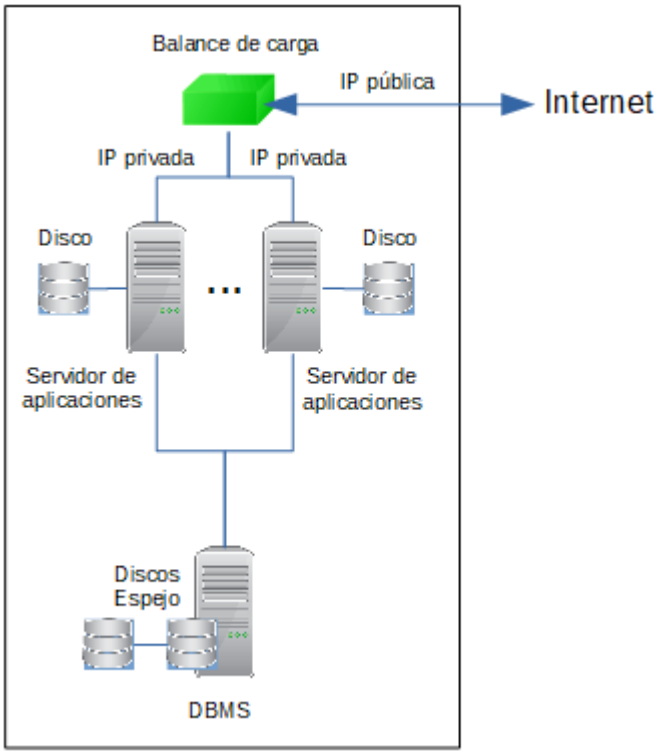
3. Revisar el artículo: Run a test failover for a single VM

- **Clase del día - 03/06/2021**

La clase de hoy vamos a ver el tema de balance de carga en la nube.

El *balance de carga* es la distribución equilibrada de carga entre un grupo de servidores (p.e. servidores web) o recursos en el *back-end* (p.e. unidades de almacenamiento).

La *carga* es el tráfico de red entrante a un recurso, por ejemplo las peticiones a un servidor, o las lecturas/escrituras a una unidad de almacenamiento.



Azure Load Balancer

El balanceador de carga en Azure opera en la capa de transporte (nivel 4) del modelo OSI, por tanto soporta los protocolos TCP y UDP.

El balanceador de carga utiliza un algoritmo de distribución haciendo el hash de cinco elementos: la IP de origen, el puerto de origen, la IP de destino, el puerto de destino y el tipo de protocolo

En Azure se puede crear dos tipos de balanceadores de carga:

Balanceador de carga público

Un balanceador de carga público mapea la dirección IP pública (IP de *front-end*) y el puerto (la IP pública y el puerto conforman un *endpoint* de Internet) a una dirección IP privada y puerto de una máquina virtual. Utilizando reglas en el balanceador de carga, es posible distribuir la carga por tipo de tráfico (HTTP, HTTPS, FTP, SMTP, SSH, RDP, MySQL, MS SQL, etc).

Balanceador de carga interno

Un balanceador de carga interno distribuye el tráfico entre los recursos que se encuentran dentro de una red virtual. Este tipo de balanceador de carga se utiliza para equilibrar la carga de las máquinas virtuales que ejecutan procesos de negocio que no son visibles al usuario externo.

La IP de un balanceador de carga interno nunca se expone como *endpoint* de Internet. En un escenario de nube híbrida, es posible conectar servidores on-premise a un balanceador de carga interno.

Escalamiento mediante balance de carga

El balance de carga permite escalar las aplicaciones e implementar servicios con alta disponibilidad dentro de una zona y a través de diferentes zonas.

Agregando máquinas virtuales al balanceador de carga los sistemas incrementan su capacidad de atender peticiones, así mismo, se evita la interrupción del servicio cuándo uno de los servidores falla.

Costo del balance de carga en Azure

En Azure se puede configurar reglas de equilibrio de carga y reglas NAT (utilizadas para mapear el tráfico entre las direcciones IP públicas y privadas). Las reglas de equilibrio de carga se cobran por número de reglas por hora. Además se cobra la cantidad de datos procesados de entrada y de salida, independientemente de las reglas.

En el caso de Standard Load Balancer no se cobra por hora si no hay reglas configuradas. Las reglas NAT son gratuitas.

Creación de un balanceador de carga en Azure

1. En el portal de Azure seleccionar "Más servicios".

2. Seleccionar "Todos los servicios".

3. En la sección "REDES" seleccionar la opción "Equilibradores de carga"

4. Seleccionar la opción "Agregar".

5. Seleccionar el grupo de recursos o bien, crear un nuevo grupo de recursos.

6. Ingresar un nombre para la instancia del balanceador de carga.

7. Seleccionar la región.

8. Seleccionar el tipo de balanceador (Interno o Público).

9. Seleccionar el SKU (Básico o Estándar).

10. Seleccionar la opción "Crear" en "Dirección IP pública".

11. Ingresar el nombre de la dirección IP pública.

12. Seleccionar la zona de disponibilidad (p.e. 1).

Una **zona de disponibilidad** es una locación física única dentro de una región. Cada zona está compuesta por uno o más datacenters equipados con su propia alimentación, refrigeración y red.

Una **región** es un conjunto de datacenters inter-conectados mediante una red de baja latencia.

13. Seleccionar "No" en "Agregar una dirección IPv6 pública".

14. Dar click en el botón "Revisar y crear".

15. Dar click en el botón "Crear".

Configuración del balanceador de carga

1. En el inicio del portal de Azure seleccionar "Todos los recursos".

2. Seleccionar el balanceador (equilibrador) de carga a configurar.

Podemos ver la IP pública creada para el balanceador de carga.

3. Para agregar máquinas virtuales al balanceador de carga seleccionar "Grupos de back-end".

4. Seleccionar la opción "Agregar".

5. Ingresar un nombre para el grupo back-end.

6. Seleccionar la red virtual.

7. Seleccionar la versión de IP (IPv4 o IPV6).

8. Dar click al botón "+Agregar" para agregar una máquina virtual al grupo back-end.

Nota importante. Las máquinas virtuales no deben tener IP pública y deben estar en la misma ubicación y red virtual que el balanceador de carga. Al crear cada máquina virtual seleccionar "Ninguno" en el campo "IP Pública" en la pestaña "Redes".

9. Marcar la máquina virtual a agregar.

10. Dar click al botón "Agregar".

11. Repetir desde el paso 8 para cada máquina virtual a agregar a grupo back-end.

12. Dar click al botón "Agregar".

13. Dar click en la campana de notificaciones para verificar que se implemente el grupo back-end de manera correcta.

Agregar un sondeo de estado

Antes de agregar reglas de equilibrio de carga es necesario crear al menos un sondeo de estado:

1. En el inicio del portal de Azure seleccionar "Todos los recursos".

2. Seleccionar el balanceador (equilibrador) de carga a configurar.

3. Seleccionar "Sondeos de estado".

4. Seleccionar la opción "+Agregar".

5. Ingresar el nombre del sondeo.

6. Seleccionar el protocolo (TCP, HTTP o HTTPS).

7. Ingresar el puerto que se sondeará.

8. Ingresar el intervalo de sondeo en segundos.

9. Ingresar el umbral incorrecto (número de errores de sondeo consecutivos que indican que el estado de la máquina virtual no es correcto).

10. Dar click en el botón "Aceptar".

Agregar una regla de equilibrio de carga

1. En el inicio del portal de Azure seleccionar "Todos los recursos".

2. Seleccionar el balanceador (equilibrador) de carga a configurar.

3. Seleccionar "Reglas de equilibrio de carga".

4. Seleccionar la opción "+Agregar".

5. Ingresar el nombre de la regla.

6. Seleccionar la versión de IP (IPv4 o IPv6).

7. Seleccionar la dirección IP de fron-end (la IP del balanceador de carga).

8. Seleccionar el protocolo (TCP o UDP).

9. Ingresar el puerto (del balanceador de carga).

10. Ingresar el puerto de back-end (el puerto en las máquinas virtuales podría ser diferente al puerto que usan los clientes para conectarse con el balanceador de carga). Este puerto deberá estar abierto en todas las máquinas virtuales dentro del grupo back-end.

11. Seleccionar el grupo de back-end (previsemente creado).

12. Seleccionar el sondeo de estado (previamente creado).

13. Dar click en el botón "Aceptar".

Agregar una regla NAT de entrada (opcional)

Una regla NAT de entrada se utiliza para reenviar el tráfico que entra al balanceador de carga hacia una máquina virtual específica. Desde luego la función del balanceador de carga es distribuir el tráfico entre diferentes máquinas virtuales, por tanto la definición de reglas NAT de entrada sería poco frecuente.

1. En el inicio del portal de Azure seleccionar "Todos los recursos".

2. Seleccionar el balanceador (equilibrador) de carga a configurar.

3. Seleccionar "Reglas NAT de entrada".

4. Seleccionar "+Agregar".

5. Ingresar un nombre para la regla.

6. Seleccionar la dirección IP de front-end.

7. Seleccionar el servicio (p.e. HTTPS)

8. Seleccionar el protocolo (TCP o UDP).

9. Ingresar el puerto (al seleccionar el servicio se inicializa el puerto por default).

10. Seleccionar la máquina virtual de destino.

11. Por omisión, el balanceador de carga envía el tráfico a la máquina virtual a través del puerto que usa el cliente para conectarse con el balanceador, si este es el caso,

seleccionar "Predeterminado" en el campo "Asignación de puertos". En otro caso seleccionar "Personalizado" para enviar el tráfico a través de otro puerto.

12. Dar click en el botón "Aceptar".

Actividades individuales a realizar

Ver las páginas:

Load Balancer

Precios de Load Balancer

Regions and Availability Zones in Azure

Ver el video:

- **Clase del día - 07/06/2021**

La clase de hoy vamos a ver el tema de replicación en la nube mediante Azure Site Recovery.

Azure Site Recovery

Azure Site Recovery (ASR) es un servicio que permite la replicación de máquinas virtuales a diferentes regiones de Azure. La configuración de la replicación se lleva a cabo directamente en el portal de Azure.

ASR permite realizar la recuperación de una aplicación multi-capa (servidor de aplicaciones+base de datos+...) ejecutando en varias máquinas virtuales.

Es posible realizar pruebas de recuperación de desastres, sin impactar el ambiente de producción y a los usuarios. Así mismo, es posible mantener las aplicaciones disponibles durante el proceso de recuperación.

Crear la máquina virtual

Crear un máquina virtual RedHat Linux 8.1 con 2 GB de RAM (B1ms) disco HDD estándar.

En este caso la máquina virtual va tener RedHat Linux 8.1 debidoa a que la replicación no está soportada para todas las versiones de los sistemas operativos. Ver la sección "Azure VM Requeriments" en

Support matrix for Azure VM disaster recovery between Azure regions

.

Replicar una máquina virtual

Después de crear la máquina virtual presionar el botón "Ir al recurso".

Seleccionar la opción "Recuperación ante desastres" en el menú que queda a la izquierda de la pantalla.

Seleccionar la región de destino si se quiere cambiar la región recomendada.

Dar click en el botón "Revisar e iniciar replicación".

Se muestra entonces la configuración del origen y el destino. Se puede ver que se creó un nuevo grupo de recursos (prueba-asr) y una nueva red virtual (prueba-vnet-asr).

Se creó un Disco administrado de réplica (en este caso un disco HD estándar), el cual es réplica del disco administrado de origen (en este caso HD estándar).

Dar click al botón "Iniciar replicación".

Revisar las Notificaciones (dar click en la campana que se encuentra en la barra superior) para verificar que se haya habilitado la replicación correctamente.

Si hay un error por falta de espacio en el disco de sistema operativo (S.O.), se puede aumentar el tamaño del disco. Este tema lo vimos en una clase previa**: Cambio del tamaño del disco de S.O.**

Actividades individuales a realizar

1. Revisar el artículo: Azure Site Recovery

2. Revisar el artículo: Inicio rápido: Configuración de la recuperación ante desastres en una región secundaria de Azure de una máquina virtual de Azure

3. Revisar el artículo: Run a test failover for a single VM

En la clase de hoy vamos a jugar un kahoot en la modalidad de "challenge".

Debido a que la plataforma Kahoot recientemente limitó a un máximo de 10 jugadores por kahoot, podrán jugar los primeros 10 alumnos que accedan al kahoot.

El kahoot se habilitará a la hora de clase, de acuerdo al horario presencial.

- Para jugar el kahoot deberán ingresar al siguiente enlace:

  Es necesario que los alumnos y alumnas ingresen su "nickname" como su nombre y apellido (por ejemplo JuanLopez), de manera que sea posible identificar a los ganadores de puntos extra.

  La hora límite para jugar este kahoot es 11:00 PM del 7 de junio.

- Cada alumno creará una máquina virtual en la nube de Azure y realizará los siguientes procedimientos que vimos en clase:

  1. Crear la imagen de la máquina virtual.
  2. Crear una máquina virtual a partir de la imagen creada.

  Se deberá entregar un reporte en formato PDF que incluya la captura de pantalla de **todos los pasos** correspondientes a los procedimientos listados anteriormente. El reporte deberá incluir además portada y conclusiones.

  El nombre de la primera máquina virtual deberá ser: el prefijo "IMG", el número de boleta del alumno, un guion y 0. Si el número de boleta del alumno es 12345678, entonces la primera máquina virtual deberá llamarse: IMG12345678-0.

  Si el número de boleta del alumno es 12345678, la máquina virtual creada a partir de la imagen deberá llamarse IMG12345678-1.

  No se admitirá la tarea si las máquinas virtuales no se nombran como se indicó anteriormente.

  No se admitirán tareas en formato RAR o en formato Word.

  Valor de la tarea: 20% (1.2 puntos de la tercera evaluación parcial).

- **Clase del día - 10/06/2021**

En clases anteriores vimos servicios de la nube al nivel de infraestructura (IaaS: Infrastructure as a Service), como son la creación de máquinas virtuales, la creación de imágenes de máquinas virtuales, la replicación de sitios (Azure Site Recovery), el respaldo de máquinas virtuales (Azure Backup) y el balance de carga (Azure Load Balancer).

La clase de hoy veremos un servicio al nivel de plataforma (PaaS: Platform as a Service), el servicio administrado de base de datos MySQL que ofrece Azure.

Platform as a Service

Cuándo una empresa instala sus sistemas en la nube requiere así mismo tener acceso a un DBMS.

Uno de los más populares manejadores de bases de datos en la actualidad es MySQL (usado por empresas de clase mundial como Google, YouTube, Facebook, Twitter, PayPal, etc.), el cual puede utilizarse en su versión gratuita (Community Server) o en su versión Enterprise Edition.

Como hemos visto en clases previas, es fácil instalar MySQL sobre una máquina virtual, crear una base de datos y acceder a ella utilizando el monitor de mysql o un programa escrito en algún lenguaje como Java, C#, Python, Node.js, entre otros.

Sin embargo, en un entorno empresarial será necesario llevar a cabo la administración de MySQL, lo cual incluye la instalación, monitoreo, respaldo de las bases de datos, recuperación ante desastres, espejo de discos, etc.

Entonces la empresa tiene dos opciones, la primera es contratar el personal que se encargue de la administración del DBMS, y la segunda contratar el servicio completo en la nube. Este servicio de DBMS completamente administrado es un ejemplo de plataforma como servicio (PaaS).

Azure Database for MySQL

Azure ofrece MySQL Community Server completamente administrado como servicio. Este servicio de nube permite a la empresa delegar la administración de la base de datos y contar con alta disponibilidad (99.99%) y escalabilidad dinámica, así como el acceso remoto a MySQL mediante conexión segura.

Creación de un servidor en Azure Database for MySQL

Para crear una base de datos MySQL se deberá ingresar al portal de Azure.

1. Seleccionar la opción "Más servicios"

2. Seleccionar la opción "Todos los servicios"

3. Seleccionar "Servidores de Azure Database for MySQL"

4. Dar click en "Agregar"

5. Seleccionar un grupo de recursos existente o crear uno nuevo.

6. Ingresar el nombre del servidor, por ejemplo: prueba-mysql

7. Seleccionar "Configurar servidor"

8. En la pantalla "Plan de tarifa" se podrá configurar las características del servidor, por ejemplo se podría bajar el número de CPU virtuales de 4 (default) a 2. En la parte derecha de la pantalla se podrá ver el resumen de precios.

9. Una vez configurado el servidor dar click en el botón "Aceptar"

10. Ingresar el nombre del usuario administrador de MysQL, por ejemplo: administrador

11. Ingresar la contraseña del usuario administrador.

12. Dar click en el botón "Revisar y crear"

(Podemos ver el costo estimado al mes 141.85 USD, 2 CPU virtuales, 100 MB de almacenamiento, 7 días de retención de copia de seguridad, redundancia local de copia de seguridad, habilitado el crecimiento automático de almacenamiento)

13. Dar click en el botón "Crear"

14. Dar click en la campana de notificaciones para revisar la implementación en curso.

15. Cuándo termine la implementación del servidor, dar click en el bóton "Ir al recurso"

Conexión al servidor MySQL

Para conectarnos al servidor de MySQL recién instalado, vamos a ejecutar el monitor de mysql en una computadora:

1. En la parte izquierda de la pantalla seleccionar "Seguridad de la conexión"

2. Dar click en la opción "Agregar IP del cliente"

3. Ingresar en "Nombre de la regla de firewall" el nombre de la regla, por ejemplo: regla1

4. Ingresar en "IP inicial" y en "IP final" la IP de la computadora cliente (la computadora que va a ejecutar el monitor de mysql).

5. Verificar en "Configuración SSL" que SSL esté habilitado.

6. Dar click en el botón "Guardar"

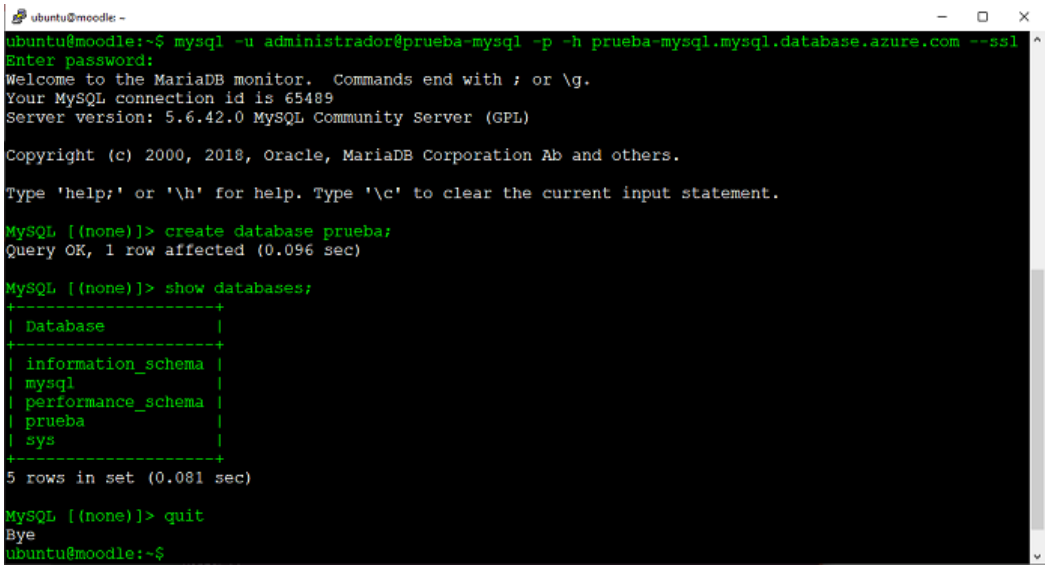
7. En el panel podemos ver el nombre del dominio del servidor, en este caso:

prueba-mysql.mysql.database.azure.com

8. Ahora podemos conectarnos al servidor de MySQL ejecutando el monitor de MySQL, en este caso se ejecuta en una computadora con Ubuntu en la cual se ha instalado previamente el monitor de MySQL:

mysql -u administrador@prueba-mysql -p -h prueba-mysql.mysql.database.azure.com --ssl

Como puede verse, la conexión con MySQL se realiza mediante SSL.



Actividades individuales a realizar

Ver los artículos:

¿Qué es PaaS?

Azure Database for MySQL

-