

Proyecto Final de la Muerte

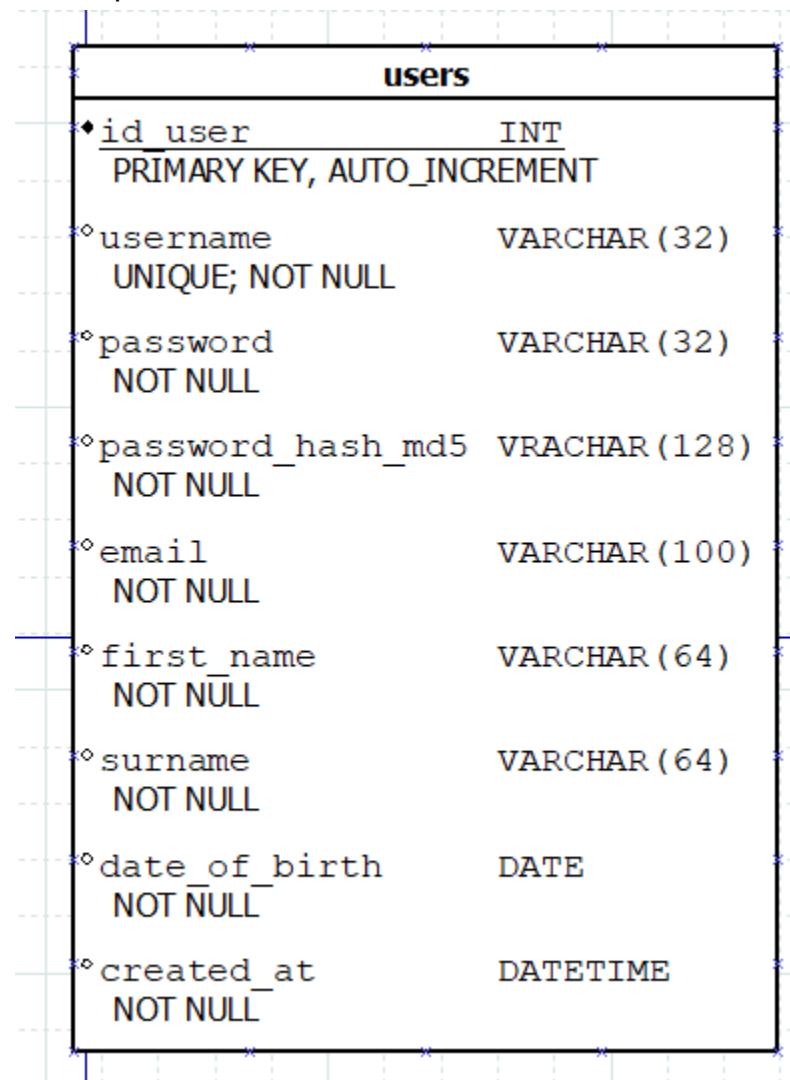


Índice

1- Primer diseño del diagrama de la base de datos:	3
2- Explicación de todos los campos del diagrama:	4
3- Script para crear la base de datos la tabla y hacer los inserts:	5
4- Script para crear los usuarios y dar los permisos:	6
5- Script para hacer el backup de la base de datos:	7
5.1- Explicación de las variables.....	8
5.2 - Explicación del script.....	8
5.3 - Archivo crontab.....	9
6- Hardware de la base de datos	10
6.1- Cálculos del espacio necesario.....	10
6.2 - Piezas para montar el servidor.....	10
Procesador:.....	10
RAM:.....	10
Discos Duros:.....	11
Configuración de discos:.....	11
Caja/Racks:.....	12
Placa Base:.....	12
7-Sistema de Clustering	13
8- Sistema de Copias de seguridad:	14
Hardware necesario:.....	14
Frecuencia y método de copia de seguridad:.....	14
Método de recuperación de fallos de disco:.....	15
Recuperación de la máquina entera:.....	15
Conclusión:.....	16

1- Primer diseño del diagrama de la base de datos:

Lo primero que hemos hecho ha sido el diagrama de la base de datos, ya que es lo más importante antes de realizar la base de datos final.



2- Explicación de todos los campos del diagrama:

ID_USER

Es el ID de la tabla users y como siempre es **INT UNSIGNED PRIMARY KEY AUTO_INCREMENT**.

USERNAME

Es el campo que guardara el nombre de usuario y será de máximo de 32 caracteres por eso es **VARCHAR(32)** el **UNIQUE** es porque no puede haber dos usernames iguales y finalmente también es **NOT NULL**.

PASSWORD_HASH_MD5

Este será el campo donde estará almacenada la contraseña de los usuarios como se puede ver no estará en texto plano, ya que podría ser inseguro además que un **MD5** es muy rápido de hacer y da una pequeña capa de seguridad este campo es **VARCHAR(32) NOT NULL**.

EMAIL

Este campo contiene el email del usuario es **VARCHAR(108)** y **NOT NULL**.

FIRST_NAME

Este campo almacena el nombre del usuario es **VARCHAR(64)** por la gente de países con nombres largos y despues es **NOT NULL**.

SURNAME

Este campo almacena los apellidos de usuario por eso es **VARCHAR(64)** y **NOT NULL**.

DATE_OF_BIRTH

Este campo almacena la data de nacimiento de la persona por eso es de tipo **DATE** y **NOT NULL**

CREATED_AT

Este campo nos indica a que hora ha sido creado el usuario para así poder tener un pequeño registro es de tipo **DATETIME** para saber a qué día y hora y minuto ha sido creado el usuario.

3- Script para crear la base de datos la tabla y hacer los inserts:

Script:

```
1  DROP DATABASE IF EXISTS usersdatabase;
2  CREATE DATABASE usersdatabase;
3  USE usersdatabase;
4
5  DROP TABLE IF EXISTS users;
6  CREATE TABLE users (
7      id_user INT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
8      username      VARCHAR(50)  NOT NULL,
9      password      VARCHAR(32)  NOT NULL,
10     password_hash_md5  VARCHAR(128) NOT NULL,
11     first_name    VARCHAR(50)  NOT NULL,
12     last_name     VARCHAR(50)  NOT NULL,
13     email         VARCHAR(100) NOT NULL,
14     date_of_birth DATE        NOT NULL
15 );
16
17 INSERT INTO users (username, password, password_hash_md5, first_name, last_name, email, date_of_birth)
18 VALUES ("root", "root", "63a9f0ea7bb98050796b649e85481845", "Mr", "Musgo", "root@gmail.com", "1990-01-01");
19
```

Lo primero que hacemos es hacer un **DROP IF EXISTS** de la base de datos para así asegurar-nos de crear-la bien, lo siguiente que hacemos es crear-la, después la seleccionamos mediante el comando **USE**, lo siguiente que hacemos es un **DROP IF EXISTS** de la tabla users, después mediante **CREATE** la creamos, e indicamos los campos que queremos que tenga y la estructura de los datos que vamos a almacenar en cada campo, por último hacemos el **INSERT INTO** en el que especificamos los campos a los que vamos a introducir el contenido y finalmente con el comando **VALUES** ponemos los datos por el orden de **INSERT INTO**, cuando terminamos debemos acabar la sentencia con un “ ; ”.

4- Script para crear los usuarios y dar los permisos.

Script:

```
1 CREATE USER 'write_update_read_user'@'localhost' IDENTIFIED BY 'enti';
2
3 GRANT INSERT, UPDATE, SELECT ON usersdatabase.users TO 'write_update_read_user'@'localhost';
```

Lo primero que hacemos es el comando **CREATE USER** después escribimos entre comillas simples el nombre de usuario en este caso '**write_update_read_user**' despues va el símbolo “ @ “ y después la IP de la máquina donde se va a usar este usuario en este caso es en la misma así que es “**LOCALHOST**”, el siguiente paso es el de dar solo los permisos de **LECTURA ESCRITURA y ACTUALIZAR**, por eso ejecutamos el comando **GRANT** y luego los permisos que queremos dar en este caso **LECTURA ESCRITURA y ACTUALIZAR**, pero los permisos no se llaman así, el de escritura se llama **INSERT**, el de **LECTURA** se llama **SELECT** y el de **ACTUALIZAR** se llama **UPDATE**, después de indicar los permisos que queremos dar debemos indicar en que base de datos y en qué tabla en este caso le hemos dado para que solo pueda en la base de datos **userdatabases** y en la tabla **users** esto se indica poniendo la base de datos después un punto y despues la tabla

(base_de_datos_que_queremos_dar_permisos.tabla)

Finalmente, debemos utilizar el comando **TO** para indicar a qué usuario queremos dar los permisos en este caso era al usuario **write_update_read_user@localhost**.

5- Script para hacer el backup de la base de datos:

Script:

```
1  #!/bin/bash
2
3
4  DB_NAME="usersdatabase"
5  BACKUP_FILE="$DB_NAME.sql"
6
7  #Compresion + fecha
8  COMPRESSED_FILE="$DB_NAME-$(date +%Y%m%d).tar.gz"
9
10 BACKUP_DIR="/home/enti/backup_DB"
11
12 #Directorio de logs
13 LOG_DIR="/var/log"
14
15 #Archivo de log
16 LOG_FILE="$LOG_DIR/$DB_NAME-backups.log"
17
18 #Volcar DB
19 mysqldump $DB_NAME > $BACKUP_DIR/$BACKUP_FILE
20
21 #Comprimir backup
22 tar -czvf $BACKUP_DIR/$COMPRESSED_FILE $BACKUP_DIR/$BACKUP_FILE
23
24 #Eliminar backup sin compresion
25 rm $BACKUP_DIR/$BACKUP_FILE
26
27 #MD5 del archivo comprimido
28 md5sum $BACKUP_DIR/$COMPRESSED_FILE > $BACKUP_DIR/$DB_NAME.md5
29
30 #Registrar la fecha y hora de la copia de seguridad en el log
31 echo "$(date +%Y-%m-%d\ %H:%M:%S) Backup realizado con éxito" >> $LOG_FILE
```

5.1- Explicación de las variables

Primero debemos declarar la variable **DB_NAME** la cual nos sirve para escoger como se llama la base de datos, la siguiente variable será **BACKUP_FILE** que será como se va a llamar el archivo sql que contendrá la base de datos, la siguiente variable será **COMPRESSED_FILE**, que será como se va a llamar el archivo comprimido, la siguiente va a ser el **BACKUP_DIR**, que será el directorio donde se va a guardar el comprimido de la base de datos, el siguiente es el **LOG_DIR** que es el directorio donde se guardará el archivo del log, el siguiente es el **LOG_FILE** que es el archivo que hará de registro del backup, una vez todas las variables ya están explicadas vamos a ver como funciona el script.

5.2 - Explicación del script

El script empieza con la sentencia:

```
mysqldump $DB_NAME > $BACKUP_DIR/$BACKUP_FILE
```

Esta sentencia utiliza el comando mysqldump el cual es el encargado de crear el archivo de recuperación de la base de datos, como podemos ejecutar el comando con la variable **DB_NAME** la cual indica a qué base de datos queremos hacer el backup, después mediante la redirección “>” indica en qué directorio se va a guardar (**BACKUP_DIR**) y como se va a llamar el archivo (**BACKUP_FILE**), la siguiente sentencia es:

```
tar -czvf $BACKUP_DIR/$COMPRESSED_FILE $BACKUP_DIR/$BACKUP_FILE
```

Esta sentencia inicia utilizando el comando **tar** con los parámetros **-czvf**, el comando **tar** crea archivos comprimidos en formato "tar" y "gzip", y los parámetros **-czvf** hacen lo siguiente:

c: crea un nuevo archivo "tar".

z: utiliza "gzip" para comprimir el archivo "tar".

v: muestra la información detallada del proceso de compresión en la pantalla (verbose).

f: indica el nombre del archivo que se está creando.

Una vez explicado este comando podemos ver que básicamente lo que hace es comprimir el archivo backup anteriormente creado a un archivo **tar.gz** y además le pone de nombre la variable **COMPRESSED_FILE** que indica a qué hora y día ha sido creada ese backup, la siguiente sentencia del script es:

```
rm $BACKUP_DIR/$BACKUP_FILE
```

Esta sentencia lo único que hace es borrar el archivo backup no comprimido mediante el comando **rm**, la siguiente sentencia es:

```
md5sum $BACKUP_DIR/$COMPRESSED_FILE > $BACKUP_DIR/$DB_NAME.md5
```

Esta sentencia también es muy simple coge el archivo de backup comprimido y le hace su archivo md5, por último tenemos la sentencia:

```
echo "$(date +%Y-%m-%d\ %H:%M:%S) Backup realizado con éxito" >>
$LOG_FILE
```

Esta sentencia hace un **echo** de la fecha y la hora en la que estamos para así con la doble redirección poder guardar este echo dentro del archivo de log para así tener un registro de que el backup ha sido realizado con éxito.

5.3 - Archivo crontab

Aquí podemos ver el archivo **crontab** el cual nos sirve para poder ejecutar el script anteriormente mostrado automáticamente cada día a las 2 de la mañana

```
enti@lluc: ~
1 # Edit this file to introduce tasks to be run by cron.
2 #
3 # Each task to run has to be defined through a single line
4 # indicating with different fields when the task will be run
5 # and what command to run for the task
6 #
7 # To define the time you can provide concrete values for
8 # minute (m), hour (h), day of month (dom), month (mon),
9 # and day of week (dow) or use '*' in these fields (for 'any').
10 #
11 # Notice that tasks will be started based on the cron's system
12 # daemon's notion of time and timezones.
13 #
14 # Output of the crontab jobs (including errors) is sent through
15 # email to the user the crontab file belongs to (unless redirected).
16 #
17 # For example, you can run a backup of all your user accounts
18 # at 5 a.m every week with:
19 # 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
20 #
21 # For more information see the manual pages of crontab(5) and cron(8)
22 #
23 # m h dom mon dow    command
24
25 0 2 * * * /home/enti/backup_script.sh
26 □
```

6- Hardware de la base de datos

6.1- Cálculos del espacio necesario

Para calcular el hardware necesario, necesitamos saber el tamaño total de la tabla de usuarios. Supongamos que cada usuario ocupa unos 200 bytes. Si la población mundial es de 7.9 mil millones de personas y queremos almacenar información de cada persona, necesitaríamos un total de 1580 terabytes de espacio libre.

6.2 - Piezas para montar el servidor

Para diseñar el hardware necesario para la base de datos, necesitamos tener en cuenta el número de usuarios, el tamaño de la base de datos y el rendimiento necesario para manejar las solicitudes. Teniendo en cuenta que necesitamos almacenar datos de toda la población del planeta, requeriremos un gran sistema el cual estará formado por:

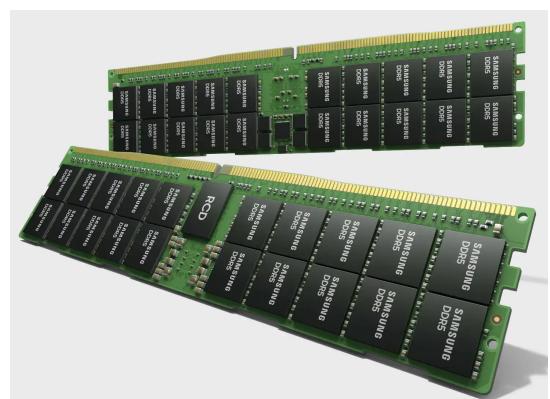
Procesador:

Para un proyecto de este tamaño, necesitamos un procesador de alta gama. Un procesador recomendado para este tipo de servidor es el Intel Xeon Platinum 8280, que cuenta con 28 núcleos y 56 hilos de procesamiento, lo que lo convierte en un procesador extremadamente potente. Además, este procesador tiene una velocidad de 2,7 GHz y una caché de 38,5 MB, o que lo hace ideal para manejar grandes cantidades de datos, su precio aproximado es de unos 11000 €. No es el mejor procesador del mercado, ya que no es de los más nuevos, pero, sin embargo, es un muy buen procesador para servidores y entre todo el mercado es uno de los mejores, calidad precio.



RAM:

Es importante tener suficiente memoria RAM para manejar grandes cantidades de datos en la base de datos, ya que en esta base de datos habrá constantemente muchos procesos a la vez eso hará que esta pueda aguantar altas tasas de demanda y muchos procesos a la vez por eso como mínimo tendríamos que tener unos 256 GB de RAM por pieza. Una buena opción es el modelo: Kingston 256GB



DDR4-2933MHz ECC Registered CL21 DIMM (KSM29RD8/256MEI) que cuesta alrededor de 1200€.

Contando que vamos a necesitar una pieza de esta por cada 4TB en total necesitaremos un total de 632 memorias RAM, haciendo que el precio total sea de 632.00€.

Discos Duros:

Para los discos duros necesitaremos un total de 1580 Terabytes, una buena opción para eso necesitaremos discos de más o menos 18TB cada uno ya los discos duros de HDD o SATA son los discos que tienen una velocidad más lenta, pero son más económicos, y de una capacidad total más grande, después los discos duros SSD, son de una capacidad total más pequeña, pero de una velocidad de transferencia de datos mucho más rápido, pero son caros así que, la mejor opción es:

Solidigm Intel D5 SSDPF2NV307TZN1 unidad de estado sólido 2.5" 30720 GB PCI Express 4.0 QLC 3D NAND NVMe



Cada uno tiene una capacidad total de 30TB si hacemos cálculos nos da un total de que necesitaremos 53 discos duros, en los que cada uno vale un total de 2300€ y el total sería de 122.000€ aproximadamente.

Configuración de discos:

En cuanto a la configuración RAID, se recomienda utilizar una configuración RAID 10 (también conocida como RAID 1+0), que combina los beneficios de RAID 1 (espejo) y RAID 0 (stripping). Esta configuración proporciona una alta capacidad de almacenamiento, así como una alta velocidad de lectura. En RAID 10, los datos se dividen en conjuntos y se duplican en discos separados, lo que permite una recuperación rápida y efectiva en caso de corrupción del disco.

Caja/Racks:

Para las cajas/racks necesitamos un modelo que sea suficientemente grande para todos nuestros otros componentes un buen modelo sería:

Modelo: Supermicro SuperChassis

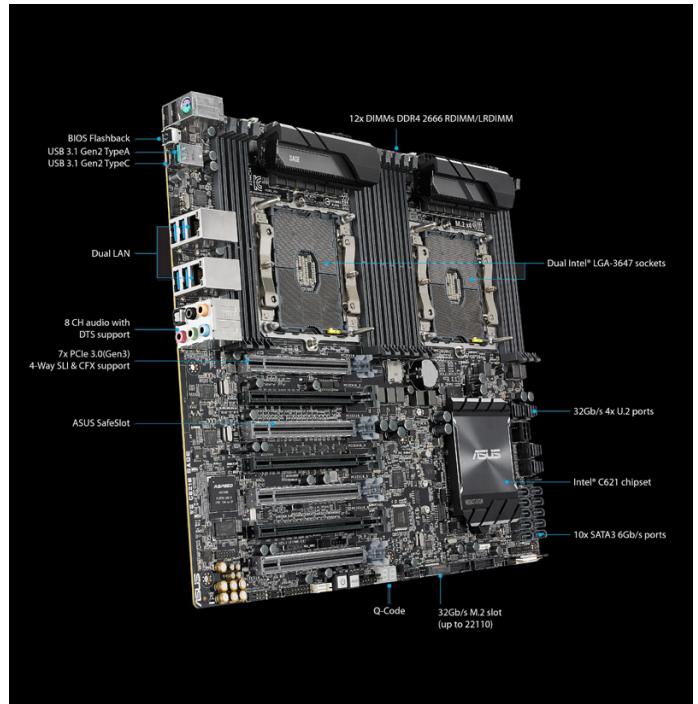
847E16-RJBOD1

Esta tiene un precio de 2500€ y tiene capacidad para unos 30 discos así que sería necesario comprar dos racks como este.



Placa Base:

Para una configuración de hardware tan potente, recomendaría una placa base de servidor de alta gama como la ASUS WS C621E SAGE. Esta placa base es compatible con los otros componentes y tiene espacio para múltiples discos duros y módulos de memoria RAM. Además, esta placa base tiene características avanzadas de administración remota, lo que la convierte en una opción sólida para servidores empresariales. Sin embargo, esta placa es una de las más caras del mercado, pero para nuestro proyecto es la más adecuada.



7-Sistema de Clustering

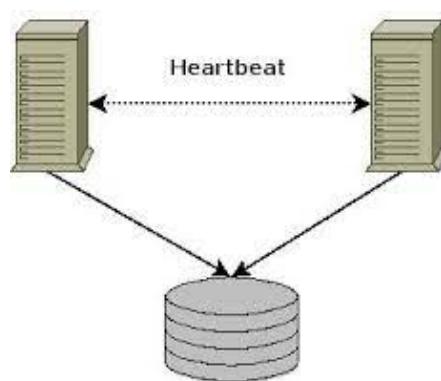
Diseñar un sistema de clustering para el servidor de base de datos implica la creación de una infraestructura que permita la distribución de la carga y la alta disponibilidad de los servicios de la base de datos. Una solución común para lograr esto es la implementación de un clúster de alta disponibilidad con varias máquinas.

Para este caso, podríamos implementar un clúster de al menos tres nodos para garantizar la disponibilidad y la redundancia en caso de fallos. Cada máquina estaría compuesta por un conjunto de discos duros, memoria RAM, procesador, y placas base que se ajusten a las especificaciones mencionadas anteriormente. Las maquinas serán de componentes un poco de menos gama que los del servidor de verdad.

Los nodos estarían conectados entre sí mediante una red de alta velocidad. El software de clustering que se utilizaría dependería del sistema operativo que se esté utilizando para el servidor de base de datos. Por ejemplo, para Linux, se puede utilizar Pacemaker y Corosync



Para garantizar la disponibilidad y la redundancia, se puede configurar el clúster de modo que, si un nodo falla, los otros nodos tomen el control y continúen proporcionando los servicios de base de datos. Además, se puede configurar el clúster para que, si un nodo está demasiado ocupado, se redirijan las solicitudes a otro nodo con más recursos disponibles.



8- Sistema de Copias de seguridad:

Hardware necesario:

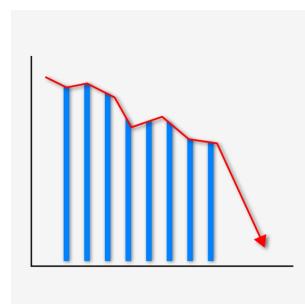
Para realizar las copias de seguridad se necesitará un dispositivo de almacenamiento externo. En este caso, se recomienda la compra de un sistema de almacenamiento en red (NAS) con una capacidad de almacenamiento suficiente para guardar las copias de seguridad.

También se necesitará una cinta de copia de seguridad para guardar una copia fuera del sitio, ya que así se pueden evitar muchísimos problemas, ya que si tenemos todas las copias en el mismo lugar si uno se infecta es muy probable que las copias de seguridad sean también afectadas.



Frecuencia y método de copia de seguridad:

Las copias de seguridad se efectuarán diariamente en un horario donde se prevea **menor actividad** en el servidor de base de datos.

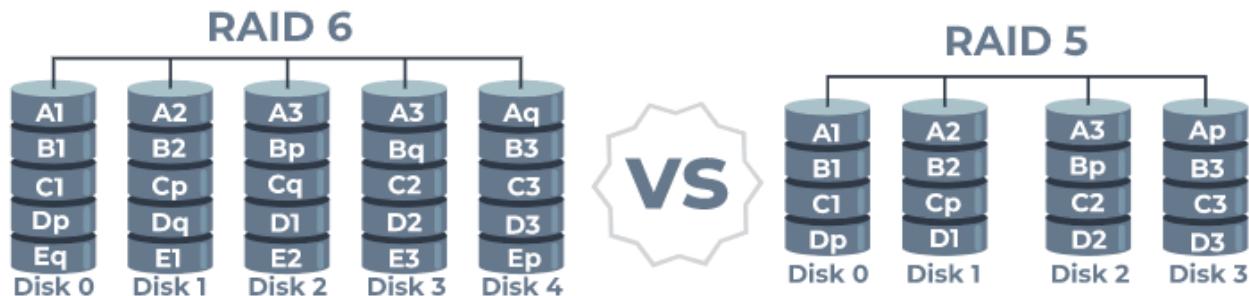


Se recomienda utilizar una herramienta de backup automatizada, como puede ser **Bacula**, que permita programar y configurar las copias de seguridad.



Método de recuperación de fallos de disco:

Si falla un disco duro, se debe reemplazar lo antes posible. Se recomienda utilizar RAID 5 o RAID 6 en la configuración de los discos duros, de manera que si uno falla, el sistema puede continuar funcionando sin problemas. La configuración RAID 5 ofrece una mayor eficiencia en el uso del almacenamiento, pero RAID 6 ofrece una mayor capacidad de recuperación en caso de fallos de disco. En cualquier caso, la configuración RAID deberá ser implementada en el hardware del servidor.



Recuperación de la máquina entera:

En caso de falla de una máquina entera, se debe contar con un plan de contingencia para recuperar la información lo más pronto posible. Para esto, se recomienda utilizar una estrategia de alta disponibilidad con un cluster de servidores. La recuperación de un servidor puede ser más rápida si se cuenta con una imagen de sistema operativo y de aplicaciones preinstalada, que permita restaurar el servidor en el menor tiempo posible.

