

O *Maximum Tolerable Downtime*, também conhecido como MTD, é uma medida do período máximo que um determinado sistema, serviço ou processo comercial pode não estar disponível antes que as consequências da indisponibilidade se tornem inaceitáveis.

O tempo de inatividade máximo tolerável para um determinado sistema ou serviço dependerá de vários fatores, incluindo a criticidade do sistema ou serviço à empresa, o impacto da indisponibilidade na empresa, a capacidade da empresa de continuar a operar sem o sistema ou serviço, e a capacidade da empresa de recuperar da indisponibilidade.

O MTD pode ser calculado através da soma do *Recovery Time Objective* (tempo médio da recuperação dos sistemas), com o *Work Recovery Time* (tempo necessário para repor os dados e aplicações e testá-los).

Assim, decidimos criar um backup parcial às 1:30PM dos dois módulos fulcrais para o funcionamento da aplicação, e um backup integral às 10:00 PM, esta solução foi dada porque não temos utilização da aplicação por volta dessa hora. (RPO)

Neste caso temos que o MTD pretendido é 20 minutos. Logo

O nosso *WRT*, *Work Recovery Time*, é o tempo que demora recuperar os dados, mais o tempo que demora a correr os scripts de testes.

Neste caso, temos que o tempo que demora a dar o *git pull*, na pior das hipóteses, mais o tempo de correr os testes é de 1 minuto.

$$20 = \text{RPO} + \text{WRT} \Leftrightarrow 20 = \text{RPO} + 1$$

O RPO é o tempo entre backups que existem para recuperar o sistema caso exista uma falha. O maior tempo neste momento é do backup parcial às 1:30PM à integral das 10:00PM no total de 510 minutos.

$$\text{MTD} > \text{RPO} + \text{WRT}$$

Para garantir que o MTD seja de 20 minutos, consideramos a implementação de cópias de segurança adicionais, como backups parciais mais frequentes para reduzir o tempo entre backups e garantir que o RPO seja inferior a 18 minutos, pois idealmente queremos que:

$$\text{MTD} > \text{RPO} + \text{WRT}$$

### **Formato NFS (Network File System)**

Criação de um espaço que permite partilhar diretórios e ficheiros com outras equipas pelo Linux através de uma rede. Para isto configurou-se um servidor que é responsável por definir com que partilha e como deixa de ser partilhado para os clientes (equipas do DEI).

#### **##Configuração do server**

#Instala *packages* relativas ao NFS

root@asist:/home/asist# **sudo apt-get install nfs-kernel-server**

#Cria um diretório para a partilha

root@asist:/home/asist# **sudo mkdir /public**

#Delega permissão de *write* and *read* aos utilizadores

root@asist:/home/asist# **sudo chown nobody:nogroup /public**

root@asist:/home/asist# **sudo chmod 777 /public**

#Define o diretório a partilhar, bem como o IP do cliente a partilhar no ficheiro */etc/exports*

root@asist:/home/asist# **sudo nano /etc/exports**

*/etc/exports*

**/public 10.9.10.0/24(rw,sync,no\_subtree\_check)**

- 10.9.10.0/24 : Ips das máquinas do DEI com mascara /24 para ter o *range* de 10.9.10.0/255

-rw : Permissão de *read* e *write*

-sync : Responde aos pedidos apenas depois de as alterações terem sido comprometidas a um armazenamento estável

-no\_subtree\_check : Assegura que os ficheiros dentro de directórios aos quais apenas o *root* tem acesso

#Torna os diretórios locais disponíveis para os clientes no *NFS* para *mount*

root@asist:/home/asist# **sudo exportfs -arvf**

**exporting 10.9.10.0/24:/public**

#Liga o servidor de NFS

root@asist:/home/asist# **sudo systemctl start nfs-kernel-server**

root@asist:/home/asist# **sudo systemctl enable nfs-kernel-server**

**Synchronizing state of nfs-kernel-server.service with SysV service script with  
/lib/systemd/systemd-sysv-install.**

**Executing: /lib/systemd/systemd-sysv-install enable nfs-kernel-server**

#Verifica os estado do servidor

root@asist:/home/asist# **sudo systemctl status nfs-kernel-server**

```
● nfs-server.service - NFS server and services
   Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled; vendor preset: enabled)
   Active: active (exited) since Wed 2022-12-28 15:20:11 WET; 2s ago
     Process: 5550 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
    Process: 5551 ExecStart=/usr/sbin/rpc.nfsd $RPCNFSDARGS (code=exited, status=0/SUCCESS)
    Main PID: 5551 (code=exited, status=0/SUCCESS)
      CPU: 8ms

dez 28 15:20:10 asist systemd[1]: Starting NFS server and services...
```

## **##Configuração do cliente**

#Instala as *packages* relativas ao *nfs-common*

client@asist:/home/asist# **sudo apt-get install nfs-common**

#Mostra os diretórios acessíveis remotamente

client@asist:/home/asist# **showmount -e 10.9.10.70**

**Export list for 10.9.10.70**

**/public 10.9.10.0/24**

#*Mount* permanente ao diretório /public

client@asist:/home/asist# **sudo nano /etc/fstab**

**/etc/fstab**

#Adicionado no ficheiro

**10.9.10.70:/public /mnt/public nfs defaults,\_netdev 0 0**

-nfs : Network File System

-defaults : rw, suid, dev, exec, auto, nouser, ect....

-\_netdev : força o systemd a considerar a unidade de *mount* uma *mount* em rede

#*Mount*

client@asist:/home/asist# **sudo mount -a**

#Para verificar os *mounts*

client@asist:/home/asist# **mount**

#Aceder ao diretório partilhado /public

client@asist:/home/asist# **cd /mnt/public/**

client@asist:/mnt/public#

## US 3 - Hugo Carvalho

Como administrador de sistemas quero que seja realizada uma cópia de segurança da(s) DB(s) para um ambiente de Cloud através de um script que a renomeie para o formato **<nome\_da\_db>\_yyyymmdd** sendo **<nome\_da\_db>** o nome da base de dados, **yyyy** o ano de realização da cópia, **mm** o mês de realização da cópia e **dd** o dia da realização da cópia.

## US 4 - Hugo Carvalho

Como administrador de sistemas quero que utilizando o Backup elaborado na US C3, seja criado um script que faça a gestão dos ficheiros resultantes desse backup, no seguinte calendário. 1 Backup por mês no último ano, 1 backup por semana no último mês, 1 backup por dia na última semana.

(Os seguintes scripts fazem parte da resolução tanto da US 3, US 4 , US 5 e US 6)

### backup.sh

```
#!/bin/bash

NC='\033[0m'
GREEN='\033[0;32m'

URL="mongodb://mongoadmin:5d57dcafb86f32d4172093e@vsgate-s1.dei.isep.ipp.pt:10337/admin"
MONGO_DAILY_DIR="/home/asist/gdrive/mongo_daily/"
MARIA_DAILY_DIR="/home/asist/gdrive/maria_daily/"
NOW=$(date "+%Y%m%d_%H_%M_%S")

logger --msgid "backup" -p local0.INFO "Downloading the MongoDB..."
echo -e "\n${GREEN}=====> Downloading the MongoDB...${NC}\n"
if mongodump --uri $URL --out $MONGO_DAILY_DIR ; then

    cd $MONGO_DAILY_DIR

    logger --msgid "backup" -p local0.INFO "Compressing the MongoDB into a ZIP file..."
    echo -e "\n${GREEN}=====> Compressing the MongoDB into a ZIP file...${NC}\n"

    if zip -rm mongo_$NOW.zip admin/ ; then
        logger --msgid "backup" -p local0.INFO "MongoDB Done."
        echo -e "\n${GREEN}=====> MongoDB Done.${NC}\n"
    else
        logger --msgid "backup" -p local0.ERROR "Failed to zip MongoDB."
        write root <<< "Failed to zip MongoDB."
    fi
else
    logger --msgid "backup" -p local0.ERROR "Failed to download MongoDB."
    write root <<< "Failed to download MongoDB."
fi

logger --msgid "backup" -p local0.INFO "Downloading the MariaDB..."
```

```

echo -e "\n${GREEN}=====> Downloading the MariaDB...${NC}\n"
if mysqldump -u root -pciTaGnrTvpWc -h vs275.dei.isep.ipp.pt --all-databases >
/home/asist/gdrive/maria_daily/dump.sql >
cd $MARIA_DAILY_DIR

logger --msgid "backup" -p local0.INFO "Compressing the MariaDB into a ZIP file..."
echo -e "\n${GREEN}=====> Compressing the MariaDB into a ZIP file...${NC}\n"

if zip -rm maria_$(date +%Y%m%d).zip dump.sql ; then
    logger --msgid "backup" -p local0.INFO "MariaDB Done."
    echo -e "\n${GREEN}=====> MariaDB Done.${NC}\n"
else
    logger --msgid "backup" -p local0.ERROR "Failed to zip MariaDB."
    write root <<< "Failed to zip MariaDB."
fi
else
    logger --msgid "backup" -p local0.ERROR "Failed to download MariaDB."
    write root <<< "Failed to download MariaDB."
fi

```

- Faz download da DB mongo para a pasta MONGO\_DAILY\_DIR
- Faz o zip da DB mongo
- Faz download da DB maria para a pasta MARIA\_DAILY\_DIR
- Faz o zip da DB maria

#### US 5

- “logger” adiciona ao ficheiro /var/log/syslog mensagens e permite enviar mensagens para o log do sistema linux
- “--msgid” atribui um id único para a mensagem
- “-p” especifica a prioridade da mensagem
- “local0.INFO”- “local0” indica a “facility” e “INFO” a “severity”
- “write” abre uma linha de comunicação que permite enviar mensagens a outros users
- “root” indica o nome de utilizador
- “<<<” redireciona a string para o root

## manage\_backups.sh

```
#!/bin/bash

NC='\033[0m'
GREEN='\033[0;32m'

MONGO_DAILY_DIR="/home/asist/gdrive/mongo_daily/"
MONGO_WEEKLY_DIR="/home/asist/gdrive/mongo_weekly/"
MONGO_MONTHLY_DIR="/home/asist/gdrive/mongo_monthly/"

MARIA_DAILY_DIR="/home/asist/gdrive/maria_daily/"
MARIA_WEEKLY_DIR="/home/asist/gdrive/maria_weekly/"
MARIA_MONTHLY_DIR="/home/asist/gdrive/maria_monthly/"

# Daily backup
/home/asist/scripts/backup.sh

#####
# Weekly and Monthly backups #
#####

# Weekly
if [ $(date +%u) == 7 ]
then
    echo -e "\n${GREEN}=====> Moving the last daily backup to weekly folder...${NC}\n"
    mv $MONGO_DAILY_DIR$(ls -t $MONGO_DAILY_DIR | head -1) $MONGO_WEEKLY_DIR
    mv $MARIA_DAILY_DIR$(ls -t $MARIA_DAILY_DIR | head -1) $MARIA_WEEKLY_DIR
fi

# Monthly
if [ $(date +%d) == 01 ]
then
    echo -e "\n${GREEN}=====> Moving the last weekly backup to monthly folder...${NC}\n"
    mv $MONGO_WEEKLY_DIR$(ls -t $MONGO_WEEKLY_DIR | head -1) $MONGO_MONTHLY_DIR
    mv $MARIA_WEEKLY_DIR$(ls -t $MARIA_WEEKLY_DIR | head -1) $MARIA_MONTHLY_DIR
fi

#####
```

```
# Delete older backups #
#####

# Daily
if [[ $(ls $MONGO_DAILY_DIR | wc -l) > 6 ]]
then
    echo -e "\n${GREEN}=====> Removing the oldest daily backup...${NC}\n"
    rm $MONGO_DAILY_DIR$(ls -t $MONGO_DAILY_DIR | tail -1)
fi
if [[ $(ls $MARIA_DAILY_DIR | wc -l) > 6 ]]
then
    echo -e "\n${GREEN}=====> Removing the oldest daily backup...${NC}\n"
    rm $MARIA_DAILY_DIR$(ls -t $MARIA_DAILY_DIR | tail -1)
fi

# Weekly
if [[ $(ls $MONGO_WEEKLY_DIR | wc -l) > 4 ]]
then
    echo -e "\n${GREEN}=====> Removing the oldest weekly backup...${NC}\n"
    rm $MONGO_WEEKLY_DIR$(ls -t $MONGO_WEEKLY_DIR | tail -1)
fi
if [[ $(ls $MARIA_WEEKLY_DIR | wc -l) > 4 ]]
then
    echo -e "\n${GREEN}=====> Removing the oldest weekly backup...${NC}\n"
    rm $MARIA_WEEKLY_DIR$(ls -t $MARIA_WEEKLY_DIR | tail -1)
fi

# Monthly
if [[ $(ls $MONGO_MONTHLY_DIR | wc -l) > 11 ]]
then
    echo -e "\n${GREEN}=====> Removing the oldest monthly backup...${NC}\n"
    rm $MONGO_MONTHLY_DIR$(ls -t $MONGO_MONTHLY_DIR | tail -1)
fi
if [[ $(ls $MARIA_MONTHLY_DIR | wc -l) > 11 ]]
then
    echo -e "\n${GREEN}=====> Removing the oldest monthly backup...${NC}\n"
    rm $MARIA_MONTHLY_DIR$(ls -t $MARIA_MONTHLY_DIR | tail -1)
fi
```

```
#####
# Sync with cloud #
#####

SYNC_DIR=$(gdrive list --query "name = 'gdrive'" --no-header | awk '{print $1}')
gdrive delete -r $SYNC_DIR

gdrive mkdir gdrive
SYNC_DIR=$(gdrive list --query "name = 'gdrive'" --no-header | awk '{print $1}')
gdrive sync upload gdrive/ $SYNC_DIR
```

- Executar o script anterior que cria dois ficheiros zip com as DB;
- Verificar se é domingo:
  - Se sim, mover os últimos backups diários para a pasta dos backups semanais;
- Verificar se é o primeiro dia do mês:
  - Se sim, mover os últimos backups semanais para a pasta dos backups mensais;
- Verificar se há mais que 6, 4 e 11 backups nas pastas dos backups diários, semanais e mensais respectivamente:
  - Se sim, remover o backup mais antigo de cada uma delas;
- Por fim, sincronizar os ficheiros resultantes.

Lista de comandos utilizados:

#### Download zipped BD files

- wget <https://fastdl.mongodb.org/tools/db/mongodb-database-tools>

#### Extract contents

- tar -zxvf mongodb-database-tools-debian11-x8

#### Copy contents to the binary folder

- cp \* /usr/bin/

#### Dump MongoDB

- mongodump --uri "mongodb://mongoadmin:5d57dcafb86f32d4172093e@vsgate-s1.dei.i  
sep.ipp.pt:10337/admin" --out "/home/asist"



### **Install ZIP and UNZIP packages**

- `apt install zip unzip`

### **ZIP folder and remove the leftover**

- `zip -rm backup.zip admin/`

### **Download GDrive**

- `wget https://github.com/BugCode1/gdrive/releases/download/`

### **Extract contents**

- `tar -zxvf gdrive_2.1.2_linux_386.tar.gz`

### **Copy contents to the binary folder**

- `cp gdrive /usr/bin/`

### **Install mariadb-client**

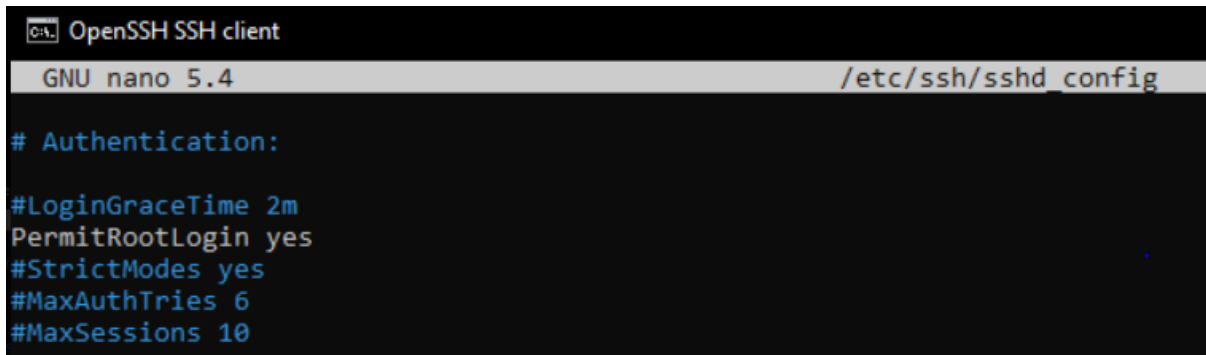
- `apt install mariadb-client`

### **Dump MariaDB**

- `mysqldump -u root -pciTaGnrTvpWc -h vs275.dei.isep.ipp.pt --all-databases > /home/asist/gdrive/maria_daily/dump.sql`

US10-Érica Lopes

Antes de criar o certificado, é preciso editar as configurações do ssh para permitir que possas te conectar com o root - nano /etc/ssh/sshd\_config



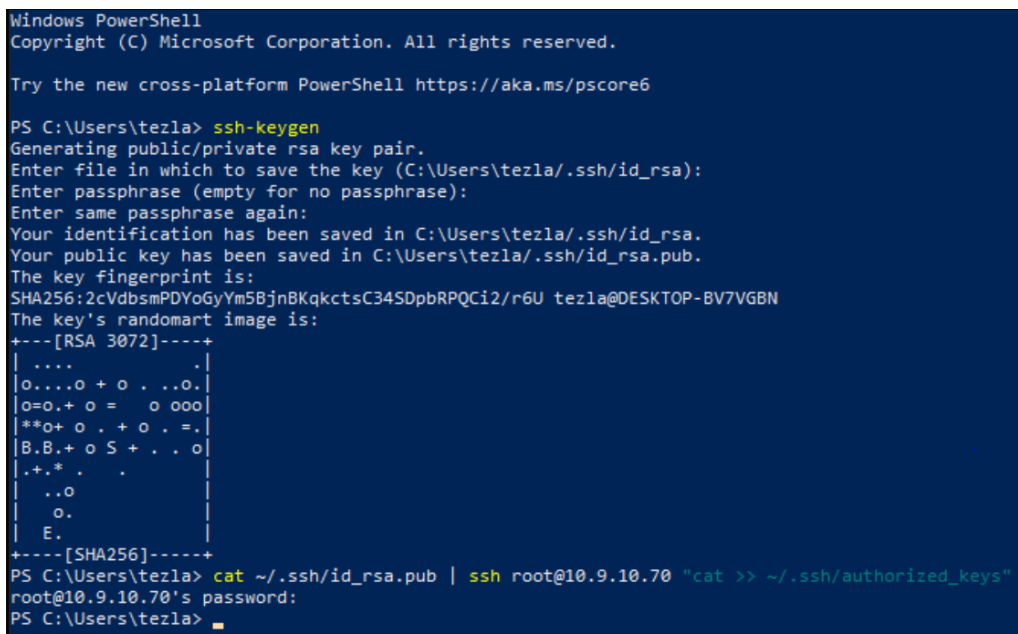
```
OpenSSH SSH client
GNU nano 5.4 /etc/ssh/sshd_config

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

Tirar o comentário da linha PermitRootLogin e definir o valor “yes”. O que permite dar login no ssh com o root.

Para atualizar as configurações, reiniciar o serviço: - service sshd restart



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\tezla> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\tezla\.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\tezla\.ssh/id_rsa.
Your public key has been saved in C:\Users\tezla\.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:2cVdbSPDYOGyYm5BjnBKqkctSC34SDpbRPQCi2/r6U tezla@DESKTOP-BV7VGBN
The key's randomart image is:
+---[RSA 3072]-----+
|      .      |
|o....o+ o . ..o|
|o=O.+ o =  o ooo|
|**o+ o . + o . =|
|B.B.+ o S + . . o|
|.+. * . .      |
| ..o           |
|  o.           |
| E.            |
+---[SHA256]-----+
PS C:\Users\tezla> cat ~/.ssh/id_rsa.pub | ssh root@10.9.10.70 "cat >> ~/.ssh/authorized_keys"
root@10.9.10.70's password:
PS C:\Users\tezla>
```

Para atualizar as configurações, reiniciar o serviço: - service sshd restart

Windows (Powershell): ssh-keygen (Gera o certificado com uma key)

cat ~/.ssh/id\_rsa.pub | ssh root@10.9.10.70 "cat >> ~/.ssh/authorized\_keys"

(Copia o certificado criado em cima para a pasta ssh do administrador da máquina do DEl)

ssh root@10.9.10.70

## US12-Érica Lopes

Como administrador de sistemas temos de garantir que em caso de necessidade os backups foram efetuados corretamente. Para isso devemos automatizar a sua reposição, validando no final o funcionamento do sistema.

```
#!/bin/bash

# Especifica o diretório onde os backups estão guardados
BACKUP_DIR="/home/asist/gdrive/mongo_daily/"

# Define a constante com a string de URI do MongoDB
MONGO_URI="mongodb://mongoadmin:5d57dcafb86f32d4172093e@vsgate-s1.
dei.isep.ipp.pt:10337/admin"

# Obtém a lista de ficheiros de backup
BACKUP_FILES=( $(ls ${BACKUP_DIR}) )

# Verifica se existem ficheiros de backup
if [[ ${#BACKUP_FILES[@]} -eq 0 ]]; then
    echo "Não há ficheiros de backup disponíveis para restauração"
    exit 1
fi

# Exibe a lista de ficheiros de backup
echo "Selecione o nome do ficheiro de backup que deseja restaurar:"
select FILE in ${BACKUP_FILES[@]}; do
    cd $BACKUP_DIR
    unzip $FILE
    echo "Unzipping file..."
    mongorestore --uri ${MONGO_URI} --drop admin/
    rm -r admin/
    mongosh ${MONGO_URI} --eval "db.getCollectionNames()"
    break
done
```