

Sprint 3

Administração de Sistemas

Turma 3DF _ Grupo 25
1200625 – Sérgio Lopes
1200628 – Tiago Freitas
1201386 – Rita Sobral
1202016 – Vasco Azevedo

Data: 08/01/2023

Índice

User Story 1.....	3
User Story 2.....	5
User Story 3.....	7
User Story 4.....	9
User Story 5.....	10
User Story 7.....	13
User Story 10.....	15
User Story 11.....	18

User Story 1

Como administrador da organização quero um plano de recuperação de desastre que satisfaça o MBCO definido na US B5.

De maneira a satisfazer o MBCO definido na User Story 5 do Sprint anterior e tendo em conta os riscos definidos na User Story 4 do Sprint anterior e o Business Impact Analysis elaborado neste Sprint elaboramos o plano de recuperação de desastres enunciado a seguir.

O plano de recuperação de desastres é um documento formal que auxilia a empresa a se preparar e responder de maneira eficaz e eficiente caso ocorram falhas e interrupções inesperadas.

Caso uma das VM's do DEI ou o Azure falhe deve ser efetuado o restauro do sistema a partir do último backup efetuado com sucesso, assegurando a rápida reparação e disponibilização dos módulos hospedados nas máquinas.

Em caso de falha na conexão à Internet, falha no fornecimento de eletricidade ou caso aconteça algum desastre natural que afete a solução desenvolvida, deve ser efetuado um processo idêntico.

Devem ser efetuados backups, mas as suas estratégias vão ser dependentes dos módulos que estamos a ponderar. Caso o serviço seja mais crítico a melhor opção serão backups incrementais diários. Para os serviços com como o MDA e o MDL o ideal seria um backup integral em 2 dias da semana, sendo um desses dias o domingo, e ainda backups incrementais a cada 12 horas.

A gestão dos ficheiros de backup está a ser realizado da seguinte maneira: 1 Backup por mês no último ano, 1 backup por semana no último mês, 1 backup por dia na última semana.

No caso de acontecer um desastre deve ser elaborado um relatório de desastre após a recuperação. Este relatório deve conter o processo detalhado na sua totalidade, incluindo a data em que aconteceu, o problema, os módulos envolvidos, as ações tomadas para resolver o problema, as medidas de prevenção que foram implementadas para evitar que o problema volte a acontecer entre outras informações que possam ser consideradas úteis na altura da elaboração.

Um relatório de desastre completo e detalhado é fundamental para garantir que a empresa se encontre preparada para lidar com eventuais desastres de maneira eficiente e eficaz.

User Story 2

Como administrador da organização quero que me seja apresentada de forma justificada a ou as alterações a realizar na infraestrutura por forma a assegurar um MTD (Maximum Tolerable Downtime) de 20 minutos.

Um MTD é um Maximum Tolerable Downtime é o tempo de inatividade máxima tolerável. Para garantir isto, é importante avaliar e identificar quais são os sistemas e serviços críticos para o funcionamento da empresa e, em seguida, implementar medidas de alta disponibilidade e tolerância a falhas para esses serviços e sistemas. Para que isso aconteça, devem surgir medidas como:

- **Implementar sistemas de monitoramento e alerta:** de forma a antever problemas no sistema, é importante garantir que temos um sistema de monitoramento que possa detetar falhas ou problemas de forma rápida e enviar alertas aos responsáveis, de modo a permitir a resolução de problemas eficientemente;
- **Implementar sistemas de backup e recuperação:** um bom plano de backup é fulcral para garantir que os dados e sistemas críticos tais como a base de dados, que é necessário ir guardando frequentemente de forma a garantir que caso dê uma falha no sistema, é possível garantir a continuação do sistema o mais rápido possível;
- **Utilizar sistemas de clustering ou balanceamento de carga:** é importante implementar sistemas que permitam a distribuição de carga entre múltiplos servidores presentes na aplicação, tornando assim possível garantir a continuidade do serviço mesmo em caso de falha de um ou mais módulos;
- **Garantir formação aos funcionários:** torna-se necessário dar formação aos funcionários para os capacitar a lidar com problemas ou falhas do sistema que possam ocorrer, incluindo como identificar e reportar bugs/problemas do sistema ou até como seguir o plano de backup.
- **Realizar testes de carga e stress:** é necessário realizar testes de carga e stress para verificar a capacidade dos sistemas e serviços críticos suportarem níveis altos de tráfego e carga, e tomar medidas para corrigir eventuais problemas identificados.

Por fim, deve se garantir que estamos preparados antecipadamente para conseguir lidar com eventuais problemas que o sistema possa vir a ter, de modo a garantir com a maior leveza possível que o mesmo volta ao normal o mais rapidamente possível, causando assim o mínimo de transtornos possíveis aos utilizadores.

User Story 3

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.

Para executar o backup às duas bases de dados, foi necessário criar dois scripts, um para a MongoDB e outra para a mysql.

Para que isso fosse possível, utilizamos o comando `wget https://fastdl.mongodb.org/tools/db/mongodb-database-tools-debian11-x86_64-100.6.1.deb` e, seguidamente instalamos o package.

Para ter acesso ao `mysqldump`, foi necessário instalar a `mariadb-client`, visto que não havia package compatível com Debian 11.

Os scripts na pasta `/home/vasco` e depois da criação dos mesmos, foi utilizado o comando `chmod 755` para dar as permissões necessárias para conseguir executar o ficheiro.

```
#!/bin/bash

export HOME=/home/vasco

TIMESTAMP=`/bin/date +%Y%m%d`
HOST=vsgate-s1.dei.isep.ipp.pt
PORT=10659
DBNAME=teste
DBPW=c871d50876a9326affb46471
US=_

DESTINATION=/home/vasco/mongodb_backup/${DBNAME}${US}${TIMESTAMP}

mkdir -p $DESTINATION

/usr/bin/mongodump --host $HOST --port $PORT --username mongoadmin --password $DBPW --out $DESTINATION

if [ -d "$DESTINATION" ]; then
    echo "Backup done"
else
    echo "Destination not found. Backup not done"
fi
```

Figura 1 - Script MongoDB

O `backupmongo` (script para fazer o backup da MongoDB), começa a instanciação das variáveis que vão ser necessárias. Foi utilizado o `TIMESTAMP` para dar adicionar a data ao nome do ficheiro.

Foi criada o directorio onde vai guardar o ficheiro com o `mkdir` e foi adicionada a flag `-p` para que, caso já exista o diretório, não haja nenhum erro.

Depois disso foi utilizado o mongodump com o host, port, username, password e destination para realizar o backup.

O if final, serve para verificar se o backup foi bem feito, visto que, se a destination for criada, então foi bem-sucedido.

```
#!/bin/sh

HOST="vs569.dei.isep.ipp.pt"
PORT="3306"
DBNAME="Basedados"
USERNAME="root"
DBPW="/fjWRZnFn40W"
TIMESTAMP="/bin/date +%Y%m%d"
US=" "
DESTINATION="/home/vasco/mysql_backup/$DBNAME$US$TIMESTAMP.sql"

/usr/bin/mysqldump --host $HOST -P $PORT -u $USERNAME -p$DBPW $DBNAME > $DESTINATION

if [ -f "$DESTINATION" ]; then
    echo "Backup done"
else
    echo "Backup failed"
    exit 1
fi
```

Figura 2 - Script Mysql

O bakcupsql (script para fazer o backup da mysql), começa da mesma maneira do outro script, com a instanciação das variáveis necessárias.

Depois disso foi utilizado o mysqldump com o host, port, username, password, nome da base de dados e destination para realizar o backup.

Tal como no script “backupmongo”, if final serve para verificar se o backup foi bem feito, visto que, se a destination for criada, então foi bem-sucedido.

Depois disto foi utilizado o comando crontab -e e foi adicionado o comando para fazer os backups diariamente, como mostra na figura representada na User story 4.

User Story 4

Quero que utilizando o Backup elaborado na US C3, seja criado um script que faça a gestão dos ficheiros resultantes desse backup. 1 Backup por mês no último ano, 1 backup por semana no último mês, 1 backup por dia na última semana.

No desenvolvimento desta US, foram respeitados os três requisitos, sendo estes: 1 Backup por mês no último ano, 1 Backup por semana no último mês, 1 Backup por dia na última semana, tendo estes critérios em mente, respondemos da seguinte maneira:

```
0 0 * * * ./backupsql.sh
0 0 * * * ./backupmongo.bash
#backup mensal do ultimo ano
0 0 1 * * ./backupsql.sh
0 0 1 * * ./backupmongo.bash
#backup semanal do ultimo mes
0 0 * * 1 ./backupsql.sh
0 0 * * 1 ./backupmongo.bash
#backup por dia na ultima semana
0 0 * * * ./backupsql.sh
0 0 * * * ./backupmongo.bash
```

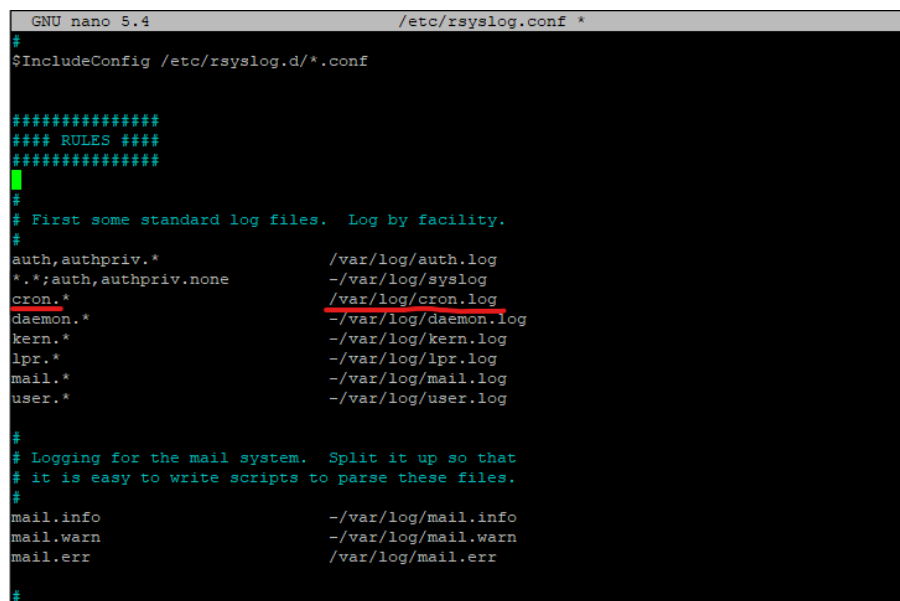
Figura 3 - Crontab

Seguindo a forma de o primeiro campo corresponder aos minutos, o seguinte à hora de um relógio no formato de 24 horas, o terceiro campo ao dia do mês, o que sucede ao mês, o quinto campo ao dia da semana e por fim o comando a ser executado.

User Story 5

Como administrador de sistemas quero que o processo da US C3 seja mantido no log do Linux, num contexto adequado, e alertado o administrador no acesso à consola se ocorrer uma falha grave neste processo.

De forma a garantir a resolução desta user story, tivemos de aceder aos scripts criados na **US3** e foi preciso adicionar uma condição. No entanto, antes de editar o que foi feito na US3, como não vinha por default com nenhum sistema de logs instalado, teve de ser instalado o pacote **rsyslog**. Para isso, foi necessário executar o comando **sudo apt install rsyslog**. A partir da instalação desse pacote, tivemos acesso ao ficheiro **rsyslog.conf** situado em **/etc**, ficheiro que serve de configuração para o **rsyslog**. Além disso, foi necessário fazer uma alteração na configuração do **rsyslog**, através do comando **sudo nano rsyslog.conf**, descomentando uma linha relacionada com os eventos de **facility cron**.



```
GNU nano 5.4 /etc/rsyslog.conf *
#
$IncludeConfig /etc/rsyslog.d/*.conf

#####
### RULES ###
#####

#
# First some standard log files.  Log by facility.
#
auth,authpriv.* /var/log/auth.log
*.;auth,authpriv.none -/var/log/syslog
cron.* /var/log/cron.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
lpr.* -/var/log/lpr.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info -/var/log/mail.info
mail.warn -/var/log/mail.warn
mail.err /var/log/mail.err

#
```

Figura 4 - Ficheiro **/etc/rsyslog.conf**

Ao retirar o comentário da linha, indicámos ao **rsyslog** que todos os eventos com **facility cron** e uma qualquer **severity** (por causa do ***** a seguir ao ponto final), devem ser registados no ficheiro **/var/log/cron.log**. Depois, executamos o comando **sudo systemctl restart rsyslog**, de maneira a reiniciar o **rsyslog** para este já assumir as alterações feitas.

Após criarmos os ficheiros **/var/log/database_backups_success.log** e **/var/log/database_backups_error.log**, executando os comandos **sudo touch**

database_backups_error.log, respetivamente (**dentro** da pasta **/var/log**), voltámos a editar o ficheiro **/etc/rsyslog.conf**, criando estas duas linhas.

```
#
# Logging events with cron & severity
# info and emerg in specified log files
#

cron.info                -/var/log/database_backups_success.log
cron.emerg                -/var/log/database_backups_error.log
```

Figura 5 - Ficheiro /etc/rsyslog.conf

Ao adicionar as linhas, indicamos ao **rsyslog** que para eventos de **facility cron** e **severity info**, para além de **loggar** no ficheiro **/var/log/cron.log**, **loggar** também no ficheiro **/var/log/database_backups_success.log**. Para eventos com **facility cron** e **severity emerg**, para além de **loggar** no ficheiro **/var/log/cron.log**, **loggar** também no ficheiro **/var/log/database_backups_error.log**. Após isto, editámos o **script** referente ao **backup** da **database MongoDB**, acrescentando:

```
if [ -d "$DESTINATION" ]; then
    echo "Backup done"
    logger -p cron.info -t "Script MongoDB Backup" "${TIME} SUCCESS! The database ($DBNAME@$HOST:$PORT) from MongoDB was backed-up!";
else
    echo "Destination not found. Backup not done"
    logger -p cron.emerg -t "Script MongoDB Backup" "${TIME} Error found! The database ($DBNAME@$HOST:$PORT) from MongoDB failed to backup!";
    exit 1
fi
```

Figura 6 - Edição do ficheiro backupmongo.bash

De seguida, verificamos se **\$DEST** é um diretório, através da condição **if[-d"\$DEST"]** (**-d** significa que vai tentar verificar se é um diretório). Se tiver sido criado **corretamente** e **não estiver vazio**, podemos considerar que o **backup** foi executado com **sucesso**. Nesse caso, o **script** executa os comandos: **logger -p cron.info -t "Script MongoDB Backup" "\${TIME} The database (\$DBNAME@\$HOST:\$PORT) from MongoDB was successfully backed-up!"**. Este executa o logger de maneira a registar nos logs do Linux eventos de **prioridade** (através da opção **-p** e tendo em conta que **prioridade = facility.severity**) **cron.info** (cron é a facility e info é a severity) com **tag** (através da opção **-t**) **"Script MongoDB Backup"**, apenas para identificar quem registou aquele evento e com a mensagem a registar, entre aspas, a seguir à **tag** - **"\${TIME} The database (\$DBNAME@\$HOST:\$PORT) from MongoDB was successfully backed-up!"** - associada ao **dia do backup (\$TIME)**, e ao **nome e localização da database remota**

(\$DBNAME@HOST:\$PORT). Este evento será **registado** nos ficheiros **/var/log/cron.log** e **/var/log/database_backups_success.log**.

Se \$DEST não for um diretório, assumimos que o **backup falhou** e executa os seguintes comandos: **logger -p cron.emerg -t "Script MongoDB Backup" "\${TIME} Error found! The database (\$DBNAME@\$HOST:\$PORT) from MongoDB failed to backup!"**. A diferença destes comandos para o de sucesso é que a **severity**, neste caso, é **emerg** (de **emergência**) e **não info**, como no caso de sucesso. Como pode acontecer uma **falha comprometedora**, é importante que a **severity** do evento **seja altíssima**, e **emerg** é a **segunda mais alta**, apenas atrás de **panic**. De resto, a mensagem será assinalada nos ficheiros **/var/log/cron.log** e **/var/log/database_backups_error.log**, por causa de ser de **facility cron** (e **severity emerg**, no caso do segundo ficheiro).

Por último, editamos o script referente ao **backup** da **database MySQL**, acrescentando:

```
if [ -f "$DESTINATION" ]; then
    echo "Backup done"
    logger -p cron.info -t "Script MySQL Backup" "${TIME} SUCCESS! The database ($DBNAME@$HOST:$PORT) from MySQL was backed-up!";
else
    echo "Backup failed"
    logger -p cron.emerg -t "Script MySQL Backup" "${TIME} Error found! The database ($DBNAME@$HOST:$PORT) from MySQL failed to backup!";
    exit 1
fi
```

Figura 7 - Edição do ficheiro backupsql.sh

As alterações feitas são iguais às alterações feitas no script de backup da database MongoDB, anteriormente explicadas. Apenas mudamos a **tag (opção -t)** para "Script MySQL Backup" e um pouco da mensagem, para ser referente à database MySQL. Desta maneira, após a execução diária, às 00h00, dos nossos backups (**cron jobs**), é sempre loggado nos ficheiros **/var/log/cron.log** e **/var/log/database_backups_success.log** o sucesso ou **/var/log/cron.log** e **/var/log/database_backups_error.log** o falhanço destes **backups**.

User Story 7

Como administrador da organização quero que me seja apresentado um BIA (Business Impact Analysis) da solução final, adaptando se e onde aplicável o(s) risco(s) da US B4.

O objetivo do Business Impact Analysis (BIA) é determinar os impactos das falhas que possam eventualmente surgir e definir medidas para minimizar ou recuperar desses impactos. Ajuda a antecipar falhas, identificar pontos fracos no negócio e desenvolver planos de contingência de maneira a lidar com os problemas.

Processos Críticos

Como definido no Sprint B, na nossa aplicação existem os módulos presentes na Tabela 1, sendo que já foram identificados os riscos que afetam o nosso sistema e determinadas as maiores vulnerabilidades, na User Story 4 do Sprint Anterior.

Módulo	Localização	Prioridade	Serviços Dependentes
MDA	MariaDB Server – DEI Cloud	2	MDL Planeamento
MDL	MongoDB Server – DEI Cloud	2	Planeamento
Visualização	Azure Web Apps	1	Todos
Planeamento	SWI Prolog Server	3	MDL

Tabela 1 - Módulos do Sistema

Conseguimos perceber a criticidade de cada um dos módulos aferindo os impactos que a sua inatividade tem para a empresa e através das consequências resultantes do não cumprimento do Recovery Time Objective (RTO), do Recovery Point Objective (RPO) e do Maximum Tolerable Downtime (MTD).

O RTO refere-se ao período máximo que um serviço deve demorar até ser restaurado após uma interrupção ou falhas enunciadas na User Story 4 do Sprint anterior, incluindo o download de dados, as reinstalações, as atualizações, etc... Resumindo é o período tolerável de inatividade, sendo um objetivo da empresa e não uma garantia.

O RPO é o período durante o qual a quantidade de dados que o sistema poderá vir a perder é tolerável para a empresa em caso de uma interrupção ou falhas enunciadas na User Story 4 do Sprint anterior. Advém do tempo máximo entre o último backup e o momento em que ocorreu a falha.

Análise de Impacto

Módulo	RTO	RPO	MTD
MDA	12 horas	12 horas	24 horas
MDL	12 horas	12 horas	24 horas
Visualização	1 hora	1 hora	2 horas
Planeamento	24 horas	24 horas	36 horas

Tabela 2 - RTO, RPO e MTD dos diferentes módulos

O módulo Master Data Armazéns (MDA) tem maioritariamente como funcionalidades a criação de armazéns e entregas, sendo estes conceitos de negócio que afetam o desempenho da empresa e que comprometem algumas funcionalidades de outros módulos e por esse motivo é necessário que o cumprimento das métricas seja estreito e que não seja muito extenso o tempo de restauro do módulo.

O módulo Master Data Logística (MDL) tem maioritariamente como funcionalidades a criação de camiões, percursos, o planeamento da frota e o registo dos utilizadores, sendo estes conceitos de negócio que afetam o desempenho da empresa. É necessário o cumprimento das métricas para que os dados pessoais dos utilizadores estejam salvaguardados e para que outras funcionalidades não sejam comprometidas.

O módulo Visualização é o serviço disponibilizado a todos os utilizadores para que consigam desempenhar de maneira eficaz e eficiente as suas tarefas, logo a falha dele comprometeria todas as operações da empresa. Por ser o módulo com maior prioridade o cliente disse que para o bom funcionamento da empresa só poderiam ser aceites pequenos períodos de indisponibilidade.

O módulo Planeamento apresenta métricas mais alargadas, devido a existir um serviço externo que assegura o planeamento da frota em caso de interrupção ou falha do Planeamento.

User Story 10

Como administrador de sistemas quero que o administrador tenha um acesso SSH à máquina virtual, apenas por certificado, sem recurso a password.

Para obter acesso à máquina virtual, apenas por certificado, sem recurso à password, devemos seguir os seguintes passos:

- Gerar uma key ssh, através do comando **ssh-keygen**;
- Copiar a chave gerada para o servidor utilizando **ssh-copy-id root@vs118** , sendo root o utilizador e vs118 o nosso server;
- Editar o ficheiro `sshd_config`, executando **sudo nano /etc/ssh/sshd_config**; Alterar o ficheiro permitindo **PubkeyAuthentication yes** e negando **PasswordAuthentication no**;
- Por fim reiniciar o ssh através do **systemctl restart ssh**;

De modo a complementar a informação dada, temos as imagens 1 a 3, sendo que a imagem 3 comprova o desenvolvimento da US.

```

root@vsl18:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:s1E9kMjRStIVPSYc7+wrJ/bSY4DFdCs0iz0wzIcMkus root@vsl18
The key's randomart image is:
+---[RSA 3072]-----+
|      +o=.o.o.o      |
|      o = +o =.=     |
|      + *..oo +.o    |
|      . ..B o.+o.    |
|      . o S* * .o    |
|      E +. = o.      |
|      . . o.         |
|      = =            |
|      . Bo.         |
+-----[SHA256]-----+
root@vsl18:~# ssh-copy-id root@vsl18
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa
.pub"
The authenticity of host 'vsl18 (127.0.1.1)' can't be established.
ECDSA key fingerprint is SHA256:1fcu2Pyq8V7dUypUxhn32+7h91zbFKKkWBXXQgqb/jk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
root@vsl18's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'root@vsl18'"
and check to make sure that only the key(s) you wanted were added.

root@vsl18:~# apt-get install sudo
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
sudo is already the newest version (1.9.5p2-3).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@vsl18:~# apt-get install nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nano is already the newest version (5.4-2+deb11u2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@vsl18:~# sudo nano /etc/ssh/sshd_config
root@vsl18:~# sudo nano /etc/ssh/sshd_config
root@vsl18:~# systemctl restart ssh
root@vsl18:~#

```

Figura 8


```

GNU nano 5.4 /etc/ssh/sshd_config
#PermitRootLogin prohibit-password
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication. Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".

```

Figura 9

```

Linux vs118 5.4.0-132-generic #148-Ubuntu SMP Mon Oct 17 16:02:06 UTC 2022 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
last login: Fri Jan 6 02:17:27 2023 from 127.0.0.1
root@vs118:~#

```

Figura 10

User Story 11

Como administrador de sistemas quero que para agilização entre as várias equipas seja criada uma partilha pública de ficheiros, formato SMB/CIFS ou NFS.

No desenvolvimento desta US inicialmente tentamos utilizar o formato NFS, no entanto estávamos a obter erro na instalação do **nfs-utils**, o que nos fez alterar para o formato SMB/CIFS.

Assim sendo, começamos com a instalação do samba na nossa máquina virtual utilizando o comando:

- **sudo apt-get install samba**

Logo após a instalação, foi criada uma pasta para a partilha dos ficheiros através do comando:

- **sudo mkdir /srv/samba**

No ficheiro de configuração do Samba **/etc/samba/smb.conf** foi adicionada uma secção para a partilha de ficheiros disponível para todos os utilizadores na rede, sem necessidade de autenticação:

```
#Criar uma partilha de ficheiros "partilha-pública" disponível a todos os users[]  
[partilha-pública]  
    path = /srv/samba  
    browsable = yes  
    guest ok = yes  
    read only = no
```

Figura 11

Por fim, inicializamos o sistema inserindo:

- **sudo systemctl start smbd**

Montando a partilha de ficheiros utilizando o seguinte comando:

- **sudo mount -t cifs //vs270.dei.isep.ipp.pt/partilha-pública /mnt -o guest**
 - ✓ vs270.dei.isep.ipp.pt, é usado para especificar o servidor onde a partilha de ficheiros está hospedada.
 - ✓ /mnt, é o local onde a partilha de ficheiros será montada no sistema, permitindo que os usuários acessem os arquivos na partilha.
 - ✓ -o guest, permite que os usuários acessem a partilha sem precisar de autenticação.