SPRINT 2

Administração de Sistemas

Miguel Oliveira 1211281

Rodrigo Castro 1220636

Rodrigo Cardoso 1221083

Mário Ribeiro 1221019

Instituto Superior de Engenharia do Porto

Indice

Índice

Us01 [Miguel Oliveira 1211281]

Problema

Implementar um processo de deployment sistemático e automatizado para o módulo Backend MastersData (MD), utilizando uma VM no DEI. O processo precisa garantir que a aplicação seja atualizada regularmente, com base nas últimas alterações na branch principal (main), e que passe por validação automatizada (teste de conformidade) antes de ser disponibilizada. A aplicação deve ser reiniciada de maneira confiável em horários de menor uso, de forma que eventuais falhas não impactem os utilizadores.

Soluções possíveis

1. Script de Deployment Manual:
    - Executar um script manualmente sempre que houver alterações significativas no repositório.
    - Limitações: não permite uma atualização consistente; depende da intervenção do administrador e aumenta o risco de falha humana; e falha com o requisito desta User story.

2. Automatização via Script e Crontab:

    Criar um script para automação do deployment que:
    - Verifique atualizações no repositório via git pull.
    - Compile o projeto .NET e instale dependências.
    - Execute os testes e, se todos forem bem-sucedidos, reinicie a aplicação.
    - Registe os logs de execução para monitoramento posterior.

    Limitações: exige configuração inicial da chave SSH no GitHub para acesso via SSH, além de configurações para o agendamento no cron.

Solução Escolhida: Automação com Script e Crontab

1. Estrutura de Pastas e Permissões:
    - Criou-se a estrutura de pastas /root/apps/MD e nela o script buildMD.sh para organizar os arquivos necessários.
    - O script recebeu permissões restritas (chmod 700) para que apenas o root possa aceder.

2. Script de Deployment (buildMDTask.sh):
   - Passos do Script:
     a. Navega até a pasta do projeto.
        cd /root/apps/MD

     b. Verifica se há novas atualizações no repositório main usando git pull.
        git pull origin main

     c. Restaura dependências e compila o projeto com dotnet restore e dotnet build.
     d. Executa os testes com dotnet test:
        - Em caso de falha nos testes, o processo é interrompido e o administrador é notificado.
        - Se os testes forem bem-sucedidos, a aplicação é iniciada ou reiniciada usando pm2 para gerenciar o processo.
        - Regista os resultados de cada execução em um arquivo de log (logs.txt).

Script escrito:

```
#!/bin/bash
project_path="/root/apps/MD/sem5pi_g55_24_25/MastersData"
project_name="MastersData"
cd "$project_path/src"

echo "********** The following information refers to the run in the date $(date) **********"

# Pulling the latest code from remote
echo -e '\n---------- Pulling code from remote... ----------'
git pull origin main

# Checking and restoring necessary packages
echo -e '\n---------- Verifying and restoring necessary packages... ----------'
dotnet restore

# Building the project
echo -e '\n---------- Building the project... ----------'
dotnet build --configuration Release

if [ $? -eq 0 ]; then
  echo -e '\n---------- The application has been built successfully! ----------'
        if pgrep -f "dotnet $project_path/src/DDDNetCore.dll" > /dev/null; then
          echo -e '\n---------- Restarting the application service... ----------'
          pm2 restart "$project_name"  # Restarts the already running instance
        else
          echo -e '\n---------- Starting the application service... ----------'
          pm2 stop "$project_name"
          pm2 start dotnet --name "$project_name" -- run "$project_path/src/DDDNetCore.dll"  # Starts a new instance if not running
        fi
  echo -e '\n---------- The application is running... ----------'
else
  echo -e '\n---------- The application could not be built. Something went wrong. ----------'
fi
```

Figura 1 - Script buildMD.sh

3. Agendamento com Crontab:
   - Configurou-se o cron para executar o script diariamente às 4h30 da manhã, de todos os dias, para minimizar impacto em utilizadores.

- Foi adicionado o agendamento no ficheiro /etc/crontab, configurado para redirecionar a saída do script para um arquivo de logs, logs.txt, localizado na mesma pasta do script (/root/apps/MD/logs.txt). Esse arquivo armazena todas as execuções realizadas pelo script, permitindo ao administrador aceder ao histórico e identificar possíveis problemas ou confirmações de sucesso nas execuções.

Ficheiro */etc/crontab* escrito:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
30 4    * * sun,mon,tue,wed,thu,fri,sat root cd /root/apps/MD &&./buildMD.sh > logs.txt
#
```

Figura 2 - Ficheiro /etc/crontab

Us02 [Mário Ribeiro 1221019]

User Story 2

Esta US tem como objetivo restringir o acesso à solução apenas aos clientes da rede interna do DEI. Por solução, entendemos a aplicação desenvolvida em Angular com a qual os utilizadores irão interagir, ou seja, o ***frontend***.

Desenvolvimento

Para restringir o acesso, permitindo que apenas os utilizadores da rede interna do DEI (conectados via cabo ou VPN) tenham acesso, teremos de configurar as regras de ***firewall*** e ajustes de rede para permitir o acesso apenas a partir de intervalos de IP específicos associados à rede interna do DEI.

Para começar temos de descobrir qual o intervalo de IP utilizado pela rede interna do DEI, esse intervalo será usado nas regras de ***firewall*** para permitir apenas o acesso interno.

Conectando à VPN do DEI e com o comando ***ipconfig***, observamos que a rede interna do DEI pertence a **10.8.0.0/16**.

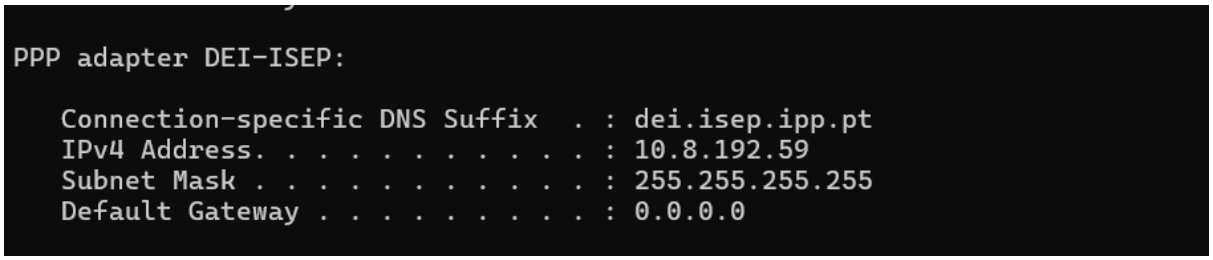

Figura 3 ipconfig

De seguida foi necessário o desenvolvimento de um script onde vão estar configuradas as regras de ***firewall***.

Foi concebido o seguinte script:

Figura 4 firewall.sh

Primeiramente, foi bloqueado todo o tráfego na cadeia ***INPUT***, que consequentemente faz com que nenhuma máquina consiga comunicar com a nossa solução. De seguida foi adicionada uma nova regra (***'-A'***) à cadeia ***INPUT*** que permite o tráfego (***'-j ACCEPT'***) na faixa de IP 10.8.0.0/16 (***-s 10.8.0.0/16***).

Posteriormente, foi incluida uma regra adicional na cadeia ***INPUT*** para permitir a comunicação com o backend da aplicação, para esta, foi considerado o IP estático 10.8.194.145 (neste caso já está coberto pela regra anterior).

De seguida, foram autorizadas as conexões SSH, que irá permitir o tráfego TCP na porta 22 (porta padrão SSH). Por fim, foram permitidos pacotes associados a conexões já estabelecidas (**tráfego TCP**) ou relacionadas a conexões já existentes (**tráfego não TCP**), que irão possibilitar respostas da ***default gateway***, garantindo desta forma o funcionamento do SSH e a resolução de nomes DNS.

De modo a tornar este script executável, é necessário dar permissões de execução ao ficheiro com o comando ***chmod +x firewall.sh***.

Por fim é necessário executar o script com ***./firewall.sh***.

Notas relevantes:

É necessário executar o script sempre que a máquina é reiniciada visto que as configurações do ***iptables*** não são persistidas entre reinicializações. Caso pretendêssemos que as mesmas fossem persistidas bastava adicionar o script ***firewall.sh*** ao ***/etc/rc.local*** para ser executado no ***boot***.

Us03 [Mário Ribeiro 1221019]

User Story 3

Esta US tem como objetivo restringir o acesso à solução apenas aos clientes da rede interna do DEI (cablada ou via VPN) através da simples alteração de um ficheiro de texto. Por solução, entendemos a aplicação desenvolvida em Angular com a qual os utilizadores irão interagir, ou seja, o ***frontend***.

Desenvolvimento

Para restringir o acesso, permitindo que apenas os utilizadores da rede interna do DEI (conectados via cabo ou VPN) tenham acesso, deve-se estar com a sessão iniciada na máquina virtual onde se encontra solução.

Começamos por criar um ficheiro de texto com o comando ***nano ipList.txt*** onde iremos colocar os endereços IP que vão ter acesso à solução.

De seguida colocamos o endereço IP da máquina e o IP referente aos clientes da rede interna do DEI (valores de referência foram os utilizados na US2).

Figura 5 - IpList

Guardamos o ficheiro e fazemos um ***reset*** à máquina virtual para não existir qualquer regra no filtro de pacotes IPv4.

Executamos o comando ***iptables -L*** para verificar as regras existentes no filtro de pacotes IPv4, é expectável que, após o realizar o ***reset***, não exista nenhuma.

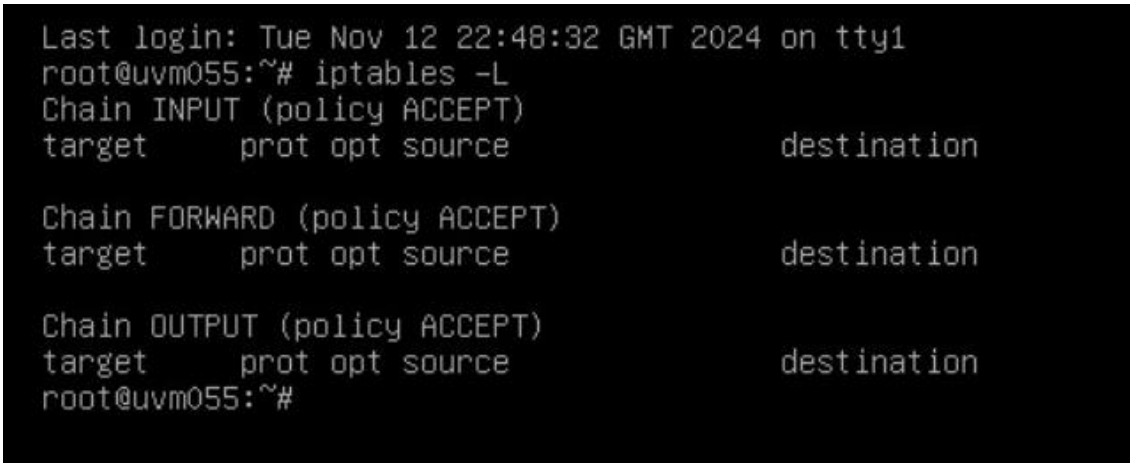

Figura 6 - IpTables

De seguida executamos a instrução ***'for ip in $(cat ipList.txt); do iptables -A INPUT -p tcp --dport 22 -s $ip/16 -j ACCEPT; done'*** para permitir o acesso aos clientes indicados no ficheiro de texto ***ipList.txt***. Este comando utiliza um ciclo ***for*** para percorrer a lista de endereços IP que constam no ficheiro ***ipList.txt*** e para cada um desses endereços, é adicionada uma regra ao ***iptables*** que permite o tráfego TCP na porta 22 (SSH) para o intervalo de endereços IP especificado.

Para que as alterações ao ***iptables*** sejam guardadas terminamos executando o comando ***/sbin/iptables-save.***

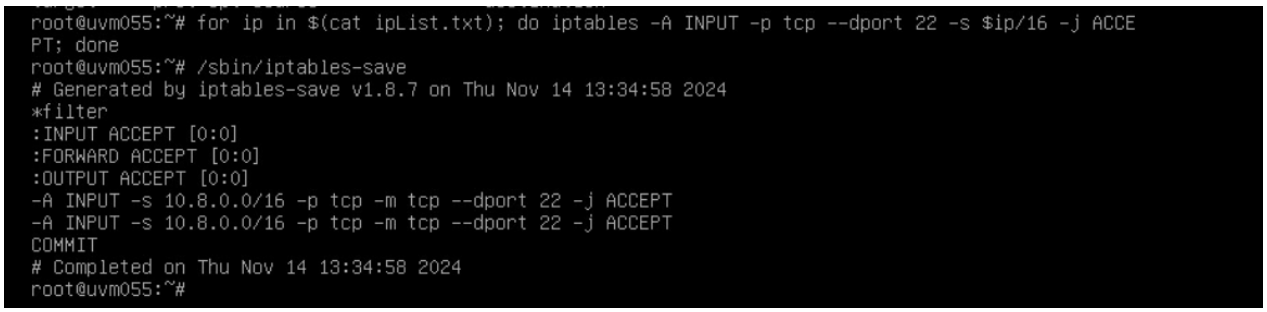

Figura 7 - IpTables-save

Com as alterações guardadas, teremos agora de executar a instrução ***'iptables -A INPUT -p tcp --dport 22 -j DROP'*** que nega o acesso a todas as outras ligações. Este comando impedirá qualquer conexão TCP na porta 22 (SSH) na entrada da firewall.

Para que esta alteração seja guardada executamos o comando ***/sbin/iptables-save*** novamente.

```
root@uvm055:~# iptables -A INPUT -p tcp --dport 22 -j DROP
root@uvm055:~# /sbin/iptables-save
# Generated by iptables-save v1.8.7 on Thu Nov 14 13:39:03 2024
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -s 10.8.0.0/16 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -s 10.8.0.0/16 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j DROP
COMMIT
# Completed on Thu Nov 14 13:39:03 2024
root@uvm055:~#
```

Figura 8 - IpTables

Para terminar verificamos se as regras foram adicionadas e estão a ser devidamente executadas com o comando ***iptables -L***.

```
root@uvm055:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  10.8.0.0/16         anywhere             tcp dpt:ssh
ACCEPT     tcp  --  10.8.0.0/16         anywhere             tcp dpt:ssh
DROP       tcp  --  anywhere            anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@uvm055:~# _
```

Figura 9 - IpTables -L

Agora apenas os clientes do DEI serão aceites e todos os outros serão descartados.

Us04 [Rodrigo Cardoso 1221083]

No âmbito desta *User Story*, como Administrador de Sistemas, quero identificar e quantificar os riscos envolvidos na solução recomendada, com base no projeto envolvente.

1. Como avaliar os riscos envolvidos

As avaliações de riscos envolvidos são normalmente representadas por uma **matriz de risco**, em que cada elemento apresenta dois fatores: a probabilidade e o impacto estimado no sistema. A medida para o risco é dada pela multiplicação dos dois fatores:

Risco = Impacto x Probabilidade

É considerado probabilidades numa escala de 1 (menos possível) a 5 (muito possível) e o impacto numa escala de 1 (marginal) a 4 (catastrófico).

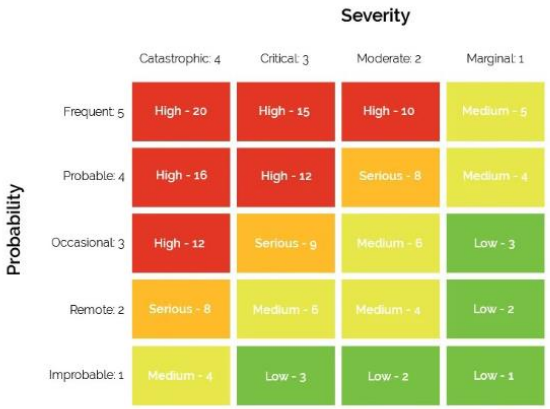

Figura 10 - Matriz de risco

No nosso caso, a matriz de risco será usada como um auxílio para a determinação de riscos associados ao dano do sistema. Foi elaborada uma **tabela** com todos as ameaças e a determinação dos seus fatores e o seu risco associado:

Ameaça	**Possibilidade**	**Impacto**	**Risco**
Avaria do Sistema	1 (Improvável)	4 (Catastrófico)	4 (Médio)
Avaria dos Servidores	2 (Remoto)	3 (Crítico)	6 (Médio)
Falhas de autenticação (Acesso indevido)	3 (Ocasional)	3 (Crítico)	9 (Sério)
Perda da Base de Dados	1 (Improvável)	4 (Catastrófico)	4 (Médio)
Leak de Dados Sensíveis	2 (Remoto)	3 (Crítico)	6 (Médio)
Vulnerabilidades do Sistema	2 (Remoto)	4 (Catastrófico)	8 (Sério)

Com base na tabela, pode-se confirmar que a maioria das ameaças apresentadas tem uma baixa possibilidade, mas terá de sempre prevenir com base no impacto envolvido.

A avaliação dos riscos é uma ferramenta necessária na definição da **BCM (Business Continuity Management)**, dado que só será possível uma medida de prevenção para cada risco assegurado com a implementação dessa mesma ferramenta.

Us05[Rodrigo Castro]

Definição do MBCO:

É o nível mínimo aceitável para assegurar a continuidade dos negócios é garantir que o sistema consiga:

1. Permitir o agendamento e a gestão de cirurgias.
2. Manter os dados essenciais armazenados e acessíveis.

Serviços Mínimos Garantidos (Funcionamento Mínimo):

Em caso de incidentes críticos, o sistema deve continuar a oferecer os seguintes serviços:

- Agendamento de cirurgias: Permitir novos agendamentos, ainda que com funcionalidades reduzidas, como registros temporários em cache.
- Consulta de dados críticos: Garantir acesso às informações indispensáveis para o funcionamento, como horários de cirurgia, alocações de sala e dados de pacientes.
- Planeamento de cirurgias: Permitir a marcação de cirurgias de forma não otimizada.

Análise dos Módulos e Funcionalidades:

1. Módulo do Servidor (Server):
    a. Impacto: Responsável pela lógica de negócio e comunicação entre os módulos. Sem ele, nenhuma funcionalidade essencial pode ser executada.
    b. Criticidade: Fundamental.
2. Base de Dados:
    a. Impacto: Armazena informações críticas, como agendamentos, pacientes e recursos. A sua indisponibilidade paralisa todo o sistema.
    b. Criticidade: Fundamental.
3. Módulo de Planeamento:
    a. Impacto: Essencial para realizar marcações otimizadas e evitar conflitos entre agendamentos.
    b. Criticidade: Fundamental.
4. Módulo de Visualização 3D:
    a. Impacto: Fornece um valor adicional, mas a sua indisponibilidade não compromete a funcionalidade central do sistema.
    b. Criticidade: Importante, mas não fundamental.

Tempo de Resposta e Recuperação:

- Deteção: A equipa será alertada em até 15 minutos após uma falha crítica.
- Diagnóstico: O problema será analisado em até 2 horas após a deteção.
- Resolução Prioritária:
    - Módulos fundamentais (Servidor, Base de Dados,Planning): Será disponibilizada uma equipa e trabalhará ininterruptamente até restaurar o funcionamento mínimo em até 6 horas.
    - Módulo 3D: Serão priorizados após a normalização dos módulos fundamentais, com prazo de restauração de até 12 horas.

Proposta de MBCO:

Garantir que, mesmo em caso de incidentes críticos:

1. O agendamento de cirurgias e a gestão de dados essenciais permaneçam operacionais de forma limitada.
2. As funcionalidades mínimas (servidor, base de dados e planeamento) sejam restauradas em até 6 horas após a falha inicial.
3. Funcionalidades secundárias, como o módulo 3D, sejam restauradas em até 12 horas após o incidente.

Justificação:

Estas medidas asseguram a continuidade do principal objetivo do negócio: o agendamento e gestão de cirurgias, minimizando os impactos em pacientes e na alocação de recursos. Essa abordagem permite:

- Reduzir interrupções em serviços críticos.
- Garantir a confiança dos utilizadores no sistema.
- Assegurar a priorização de recursos na resolução de falhas mais impactantes

Us06 [Miguel Oliveira 1211281]

Objetivo

Como administrador do sistema, o objetivo é implementar uma estratégia de backup que minimize o RPO e o WRT para a base de dados MySQL. Levando em consideração que a aplicação no backend entra em manutenção às 4h30 da manhã, o backup pode ser realizado nesse horário, aproveitando o período de baixa utilização. No entanto, para garantir que o sistema esteja sempre protegido, será feito um backup completo semanal, com backups incrementais diários durante a semana.

A estratégia será estruturada da seguinte forma:

- **RPO (Recovery Point Objective):** O objetivo é ter um RPO de 24 horas, ou seja, o base de dados poderá ser restaurado até 24 horas após a falha. Isso atende bem às necessidades do sistema, considerando o fluxo de trabalho e o uso do sistema.
- **WRT (Work Recovery Time):** O WRT será otimizado ao realizar backups incrementais durante a semana e backups completos aos domingos. Isso permitirá uma recuperação mais rápida durante a semana (em casos de falha), enquanto o backup completo aos domingos garante a recuperação total dos dados com um custo de tempo maior apenas uma vez por semana.

Estratégia de Backup

1. **Backup Completo Semanal (Domingo):** Aos domingos, será realizado um backup completo da base de dados MySQL. Esse backup inclui todos os dados do base de dados, garantindo que, em caso de falha, a recuperação seja mais rápida, uma vez que um único backup completo estará disponível.
2. **Backup Incremental Diário:** De segunda a sábado, será realizado um backup incremental. Este tipo de backup registra apenas as alterações feitas no base de dados desde o último backup. Isso otimiza o tempo de backup e reduz o espaço necessário para armazenar os backups.
3. **Agendamento do Backup:** Os backups serão feitos diariamente às 4h30 da manhã, durante a janela de manutenção da aplicação.

Implementação

A seguir, vou detalhar como implementar o script de backup, considerando o MySQL como sistema de base de dados. Para fazer esta implementação, começou-se então por instalar as ferramentos do mysql na máquina virtual,que não será aqui descrito o processo pois não é diretamente importante para a documentação da US.

Criou-se então uma script para conseguirmos fazer as cópias de segurança desejadas. Para o fazer, foi utilizado o comando "nano /etc/mysql-backup.sh".

```
!/bin/bash

# Caminhos de diret      rios para backup
BACKUP_DIR="/mysql_db_backup"
OLD_BACKUP_DIR="$BACKUP_DIR/old"
MYSQL_BACKUP_DIR="/root/mysql"

# Verifica se o diret      rio de backups antigos existe, se n      o, cria-o
if [ ! -d "$OLD_BACKUP_DIR" ]; then
  mkdir -p "$OLD_BACKUP_DIR"
fi

# Verifica se o dia da semana        domingo (0 para domingo)
DAY_OF_WEEK=$(date +%u)
if [ "$DAY_OF_WEEK" -eq 7 ]; then
  # Se for domingo, move os backups antigos para o diret      rio "old"
  mv $BACKUP_DIR/* $OLD_BACKUP_DIR/
  echo "Backup total ser       realizado, pois        domingo."
  FULL_BACKUP=true
else
  FULL_BACKUP=false
fi

# Cria o diret      rio de backup caso n      o exista
if [ ! -d "$MYSQL_BACKUP_DIR" ]; then
  mkdir -p "$MYSQL_BACKUP_DIR"
fi

# Definindo data para o arquivo de backup
DATE=$(date +'%Y-%m-%d_%H-%M-%S')
BACKUP_FILE="$MYSQL_BACKUP_DIR/mysql_backup_$DATE.sql"

# Definindo o arquivo de log para o backup incremental
SNAPSHOT_FILE="$BACKUP_DIR/mysql_backup_snapshot.snap"

# Fazendo backup completo ou incremental dependendo do dia da semana
if [ "$FULL_BACKUP" = true ]; then
  # Backup completo usando mysqldump
  mysqldump -u root -p --all-databases > $BACKUP_FILE
else
  # Backup incremental usando o mysqldump com registros de modifica            es
  mysqldump -u root -p --all-databases --single-transaction --flush-logs > $BACKUP_FILE
fi

# Mudando permiss      es para que apenas root tenha acesso
chmod -R 700 $MYSQL_BACKUP_DIR

# Comprimindo o arquivo de backup com tar (gzip) e utilizando snapshot incremental
TAR_FILE="$BACKUP_DIR/mysql_backup_$DATE.tar.gz"
tar -czpf $TAR_FILE -g $SNAPSHOT_FILE $MYSQL_BACKUP_DIR

# Limpando arquivos tempor      rios de backup
rm $BACKUP_FILE

echo "Backup conclu      do em $TAR_FILE"
```

Figura 11 - ficheiro /etc/mysql-backup.sh

Explicação do Script:

- **Criação de Diretórios**: O script verifica se os diretórios necessários para os backups existem, caso contrário, cria-os.
- **Verificação de Domingo**: O script verifica se é domingo (dia 7) e, se for, realiza um backup completo. Caso contrário, realiza um backup incremental, garantindo que apenas as mudanças feitas desde o último backup sejam armazenadas.
    - **Backup Completo vs. Incremental**:
        - backup completo (mysqldump --all-databases) é feito aos domingos.
        - backup incremental é feito durante a semana, utilizando a opção --flush-logs para garantir que apenas os dados modificados desde o último backup sejam capturados.
- **Compreensão e Snapshots**: O comando tar é usado para comprimir o backup e manter um snapshot incremental, permitindo que o sistema saiba o que foi alterado entre os backups.
- **Permissões**: Após a criação do backup, as permissões do diretório são ajustadas para garantir que apenas o root tenha acesso completo.

Agendamento do Backup com Cron

Para agendar a execução do script, foi editado o arquivo de crontab com o comando:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
30 4    * * sun,mon,tue,wed,thu,fri,sat root cd /root/apps/MD &&./buildMD.sh > logs.txt
30 4    * * *   root    /etc/mysql-backup.sh
#
```

Figura 12 - ficheiro crontab US6

Este passo é semelhante ao usado na User Story 1.

A estratégia proposta de backups diários incrementais e backups completos aos domingos visa minimizar o RPO e o WRT. Com esta configuração, o RPO será de 24 horas e o WRT será reduzido, uma vez que as falhas podem ser recuperadas rapidamente com os backups completos aos domingos, e os backups incrementais garantirão que o espaço de armazenamento e o tempo de backup sejam minimizados durante a semana.

Us07 [Rodrigo Cardoso 1221083]

No âmbito desta User Story, como Administrador de Sistemas, pretende-se definir uma pasta pública para todos os utilizadores registados no sistema, onde poderão ler o que foi lá colocado.

1. Criação da pasta pública

Foi criado uma pasta com o nome "public" no sistema Linux, com o comando:

mkdir /public

Para tornar a pasta pública, as permissões do diretório terão de ser alteradas. Poderá ser feito com o seguinte comando:

chmod 755 /public

Onde correspondem as seguintes permissões (representadas na forma octal):

- **7 (leitura, escrita e execução)** para o proprietário do diretório (neste caso a *root*)
- **5 (leitura e execução)** para os grupos e utilizadores do sistema

Quando é atribuído as permissões ao diretório, as permissões dos ficheiros e/ou pastas que estão presentes serão as mesmas da pasta principal.

2. Inserção do ficheiro dentro da pasta

Foi inserido um ficheiro dentro da pasta pública com os seguintes comandos:

cd /public

nano test.txt

Foi escrito um texto dentro do ficheiro criado e depois guardado.

Figura 13 - Atribuição de texto ao ficheiro criado

3. Testes de escrita e leitura do ficheiro pelos utilizadores

Foi iniciado sessão como um utilizador que não é administrador no sistema (no nosso caso foi o luser1).

O utilizador **conseguiu aceder à pasta pública** e assim conseguiu abrir o ficheiro presente na pasta.

Com a abertura do ficheiro, tentou alterar o conteúdo presente no ficheiro e verificou-se no seguinte erro:

Figura 14 - Tentativa de alteração do ficheiro presente na pasta partilhada

Com esse erro conseguiu-se provar que as permissões atribuídas anteriormente à pasta estão a funcionar e, como dito anteriormente, o utilizador **só irá conseguir ler** o conteúdo presente nos ficheiros/pastas dentro da pasta pública.

A criação de uma pasta pública em que todos os utilizadores poderão ler o conteúdo presente é importante na segurança da pasta, para que permite o **acesso geral** a todos os utilizadores, mas **não poderão escrever** o que está presente dentro do diretório.

Us08 [Rodrigo Castro]

Objetivo do Script

O script desenvolvido tem como objetivo identificar utilizadores que tentaram fazer login de forma errada mais de três vezes num sistema Linux. Ele lê o arquivo de log de autenticação, filtra as falhas de login e grava as informações dos utilizadores com mais de três tentativas falhadas num arquivo de saída.

Descrição Detalhada

1. **Definição dos Arquivos**

O script começa com a definição de dois arquivos:

- **Arquivo de log** (/var/log/auth.log): Contém as tentativas de login realizadas no sistema.
- **Arquivo de saída** (failed_attempts.txt): Será utilizado para armazenar os resultados do script, ou seja, os utilizadores que tentaram fazer login mais de três vezes de forma errada.

- **Mensagem Inicial**

O script exibe uma mensagem a indicar que está à procura de tentativas de login falhadas.

3. **Filtragem e Contagem das Tentativas Falhadas**
   a. O comando grep é utilizado para procurar por linhas que contêm o texto "FAILED LOGIN", indicando tentativas de login falhadas.
   b. O comando awk é usado para extrair o 12.º campo de cada linha, que corresponde ao nome do utilizador que tentou fazer login.
   c. Em seguida, o comando sort organiza as entradas de utilizadores, e o comando uniq -c conta quantas falhas cada utilizador teve.
4. **Processamento dos Resultados**

O script então percorre cada linha da saída utilizando um loop while, lendo o número de falhas ($count) e o nome do utilizador ($user).

Dentro do loop, o script verifica se o número de falhas é maior do que 3. Se for, ele regista o nome do utilizador e o número de falhas no arquivo de saída (failed_attempts.txt). Também exibe uma mensagem a indicar que a informação foi registada.

5. **Mensagem Final**

Ao final da execução, o script exibe uma mensagem a informar que os resultados foram guardados no arquivo de saída.

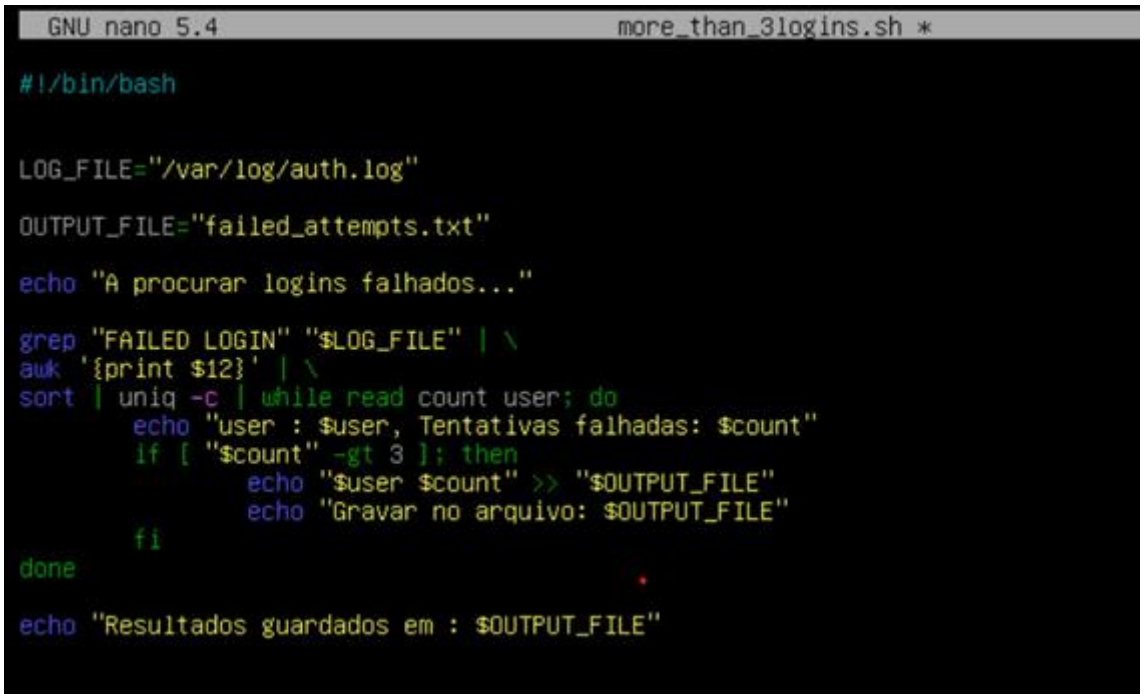

Figura 1Bash Script que encontra os users com mais do que 3 logins incorretos

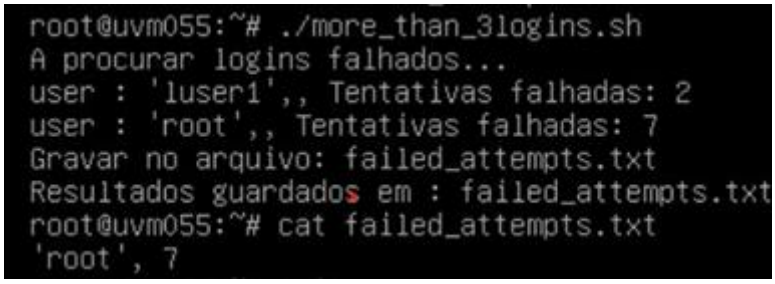

Figura 1Bash Script que encontra os users com mais do que 3 logins incorretos