Relatório de ASIST

Sprint 2

Turma 3DG – Grupo 038

1190718 – João Beires

1190782 – José Soares

1190811 – Lourenço Melo

1191419 – José Maia

Data: 03/12/2022

User Story 2

Para conseguirmos ter um ponto de situação para resolver esta “User Story”, verificamos primeiro o nosso “Ip address” e saber qual é a nossa máscara de rede. O comando utilizado foi: ifconfig | grep -i mask. Em que utilizamos o comando config e procuramos na mensagem de resposta desse comando onde se encontra escrito “mask”.

```
root@uvm038:~/backup# ifconfig | grep -i mask
        inet 10.9.10.38  netmask 255.255.0.0  broadcast 10.9.255.255
        inet 127.0.0.1  netmask 255.0.0.0
```

Vamos eliminar todas as regras que possam estar previamente feitas com o comando “iptables -F”. A “option -F” corresponder a dar flush que elimina todas as regras uma a uma.

```
root@uvm038:~/backup# /sbin/iptables -F
root@uvm038:~/backup# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
```

Como podemos visualizar com o comando “iptables -S” estão apenas as regras “default” do iptables. A “option -S” corresponde a “—list-rules”, ou seja, mostra todas regras.

De seguida vamos bloquear todo o tráfego da porta 3000. Esta porta é a porta da aplicação de “Node.js” que irá correr nesta máquina. Para poder realizar o que foi descrito iremos correr o comando “iptables -A INPUT -p tcp –destination-port 3000 -j DROP”. A option “-A” quer dizer append, ou seja, adicionar, o “INPUT” destina-se a todo o tráfego que a máquina recebe. O parâmetro “-p” serve para definir o protocolo que neste caso é “tcp”, “—destination-port” serve para definir a porta que se quer aplicar a regra e “-j” dá jump.

```
root@uvm038:~/backup# iptables -A INPUT -p tcp --destination-port 3000 -j DROP
```

Após este comando iremos aceitar todas as conexões na porta 3000, apenas aos clientes da rede interna do DEI seja por forma cablada ou via VPN. O comando “iptables -A INPUT -p tcp –dport 3000 -m iprange –src-range 10.9.1.1-10.9.255.255 -j ACCEPT”. Com o “iprange –scr-range” serve para podermos definir uma range de “ips” de forma mais especifica sem utilizar as máscaras.

```
root@uvm038:~/backup# iptables -A INPUT -p tcp --dport 3000 -m iprange --src-range 10.9.1.1-10.9.255
.255 -j ACCEPT
```

O resultado será:

```
root@uvm038:~/backup# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -p tcp -m tcp --dport 3000 -j DROP
-A INPUT -p tcp -m tcp --dport 3000 -m iprange --src-range 10.9.1.1-10.9.255.255 -j ACCEPT
```

Isto foi feito desta maneira, pois devem ser primeiro realizadas as regras mais gerais ou bloqueadoras e depois descritas as regras mais especificas.

Posteriormente a testar as regras, poderíamos instalar o “iptables-persistent” e depois utilizar o “iptables-save” para guardar após um reboot.

User Story 4

A Matriz de Riscos ou Matriz de Probabilidade e Impacto é uma ferramenta de gerenciamento de riscos que permite de forma visual identificar quais são os riscos que devem receber mais atenção

Figura 1 - Matriz de Risco

Ameaça	Probabilidade	Consequência	Risco
Falha energética	3	1	Médio - 3
Avaria num dos componentes informáticos	2	2	Médio - 4
Incêndio no data center	1	4	Médio - 4
Perda de informações essenciais	3	2	Médio - 6
Reclamação de usuários por bugs	3	2	Médio - 6
Requisitos do projeto mal-entendidos	2	4	Séria - 8
Cronograma fora da realidade	4	3	Alta - 12
Falha da VM do DEI	3	3	Séria - 9
Ataques DDoS e DoS	2	4	Séria - 8

Uma questão muito presente nos riscos de uma solução para um projeto é a perda de informações guardadas. Para isso, seriam necessários Backups.

Os Backups e a sua frequência variam com o sistema que estamos a ponderar. No caso de um serviço mais crítico, como é o caso do SPA, poderá ser necessário o uso de Backups incrementais diários.

Para os serviços com menor prioridade, seria ideal um backup integral uma vez por semana e ainda um backup incremental diário.

Para outros serviços com maior prioridade, seria talvez ideal um backup integral em 2/3 dias da semana e ainda backups incrementais de 12 em 12 horas (2 vezes por dia).

User Story 5

MBCO (Minimum Business Continuity Objective), é o nível mínimo de serviços e/ou produtos aceitável para a organização atingir os seus níveis negócio durante um desastre ou rutura. No caso da nossa aplicação, teremos de nos focar no nível mínimo de serviços que necessitaremos de continuar a fornecer.

A missão da nossa organização é gerir a **distribuição da maneira mais rentável e fiável possível de entregas entre armazéns** utilizando uma frota de camiões elétricos da empresa *"EletricAcme, S.A"*. Os **serviços essenciais** da nossa aplicação são: a criação das melhores rotas de entregas para a empresa e a visualização de todos os dados acerca das entregas, rotas, camiões e armazéns.

Existindo algum **problema com a base de dados** do programa (ex. hack, avaria, sobreaquecimento), haverá vários problemas, pois sem os dados nela guardados seria impossível criar as rotas. Mal o problema seja detetado uma **equipa de Resposta a Incidentes** será chamada, esta equipa irá ter como **principais funções restaurar e avaliar os danos**. Os dados de "*backup*", conseguidos através de cópias periódicas de segurança, guardados numa base de dados separada (já pensada para este tipo de situações), ou até mesmo em caso de avaria desta mesma os dados em forma física (papel), serão requeridos para restaurar o mais rápido possível o funcionamento correto da aplicação. A equipa trabalhará em turnos de 4 horas durante 16 horas, se necessário, para reinserir os dados na plataforma e responder a pedidos de ajuda por parte da empresa.

À medida que os dados são inseridos, o **MBCO** será garantir o funcionamento, mesmo que ainda sem 100% dos dados, do serviço de consulta de dados sobre camiões, entregas e armazéns já inseridos na plataforma até ao momento. Não faria sentido a criação de rotas de entregas sem a totalidade dos dados pois poderia fazer com que certas rotas fossem criadas sem todas as entregas desse dia, causando um grande transtorno á *"EletricAcme, S.A".*

Não iremos continuar com a manutenção normal da aplicação enquanto a situação não se encontrar regularizada, todos os recursos serão utilizados para recuperar o funcionamento total da aplicação.

User Story 6

Para conseguir resolver esta “User Story” com especial atenção para minimizar o RPO (Recovery Point Objective) e o WRT (Work Recory Time), pensamos em descarregar os repositórios de forma periódica. A frequência com que essa atividade irá ser executada será todos os dias às 00:00.

Esta decisão foi tomada por uma reflexão, que levou em conta o trabalho diário de 5 alunos no projeto integrador, ou seja, se cada aluno trabalhar cerca de 3 horas por dia, no final do dia seria 15 horas de trabalho conjunto. Caso algo acontecesse seria possível descarregar a versão que irá estar na máquina virtual assim evitando um retrocesso grande no trabalho que já tinha sido realizado. Se este “backup” fosse realizado semana a semana seria um total de 75 horas caso os alunos trabalhassem no projeto apenas nos dias da semana. Com estes valores chegamos à conclusão, de que a primeira opção seria bastante válida e positiva.

Assim para conseguir satisfazer as necessidades acima descritas, pensamos em criar um script capaz de criar uma pasta e descarregar os repositórios do “BitBucket” e remover a versão anterior (desatualizada). Deste modo, na máquina virtual Linux “uvm038” ter sempre um “backup” referente à 00:00 desse dia.

Primeiramente para termos a possibilidade de descarregar um repositório na linha de comandos foi necessária instalar o git na máquina Linux “apt-get install git”.

```
root@uvm038:~/backup# apt-get install git
A ler as listas de pacotes... Pronto
A construir árvore de dependências... Pronto
A ler a informação de estado... Pronto
git is already the newest version (1:2.30.2-1).
0 pacotes actualizados, 0 pacotes novos instalados, 0 a remover e 57 não actualizados.
root@uvm038:~/backup#
```

Após isso criamos o script chamado “script.sh”.

Script.sh

```
 GNU nano 5.4                                  script.sh
#!/bin/sh
rm -r /root/backup/repos.tgz
mkdir /root/backup/clone
cd /root/backup/clone
git clone https://LourencoMelo:                    @bitbucket.org/zesoares/lei22_23_3dg_g38_logisti>
git clone https://LourencoMelo:                    @bitbucket.org/zesoares/lei22_23_3dg_g38_gestao.>
git clone https://LourencoMelo:                    @bitbucket.org/zesoares/lei22_23_3dg_g38_spa.git
cd /root/backup
tar zcvf repos.tgz ./clone/
rm -r /root/backup/clone
```

A primeira linha remove de forma recursiva com a flag "-r" o ficheiro "repos.tgz" e os seus subdiretórios (versão antiga comprimida). Após esse comando é criado o diretório clone com o comando "mkdir /root/backup/clone" para poder agrupar todos os 3 repositórios num só diretório para facilitar processos que irão ser feitos posteriormente. Para que isso seja possível mudamos para o repositório clone com o comando "cd /root/backup/clone".

Dentro desse repositório utilizamos o comando "git clone" para clonar os repositórios do projeto integrador que fazem parte dos módulos necessários. A parte do print screen que aparece a branco serve para esconder a "App password" do "Bitbucket" pessoal, por questões de segurança.

De seguida, voltamos para o diretório backup para poder comprimir o diretório "clone". O comando "tar zcvf repos.tgz ./clone/" irá comprimir o diretório "clone" num ficheiro chamado "repos.tgz". O argumento "c" representa a criação, "v" para mostrar mais informação, ou seja, verboso ou palavroso, "f" para ficheiro e "z" para filtrar o arquivo através do gzip.

Depois eliminamos o diretório "clone" para poupar espaço de memória na máquina.

Anteriormente tínhamos usado este comando "tar cvf repos.tar ./clone/", mas ao utilizar o comando "du -h" reparávamos que a compressão não criava diferenças grandes em tamanho do ficheiro, então procuramos maneiras melhores de comprimir diretórios grandes. Na imagem abaixo conseguimos ver como o comando que utilizamos no script foi bastante mais eficiente a comprimir o diretório "clone". Este comando mostra o "disk usage" de um ficheiro ou diretório.

```
938M    clone/
root@uvm038:~/backup# du -h repos.tgz
677M    repos.tgz
root@uvm038:~/backup# du -h repos.tar
929M    repos.tar
```

Então como foi referido anteriormente, precisamos de programar que a máquina de Linux para correr este script todos os dias às 00:00. Utilizamos o comando "crontab -e -u root", em que "-

e” representa edit e “-u” user. Pois inicialmente, apenas correndo o comando “crontab -e” é nos dito que o root não tem crontab.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 0 * * * /root/backup/script.sh
```

Para a demonstração da “User Story 6”, a imagem abaixo mostra que o ficheiro comprimido foi criado alguns minutos depois da 00:00, devido ao tempo de clonar os repositórios de comprimi-los.

```
root@uvm038:~/backup# ls -l
total 693124
-rw-r--r-- 1 root r0oot 709749296 nov 26 00:02 repos.tgz
-rwxr-xr-x 1 root r0oot       465 nov 25 19:41 script.sh
```

Caso seja necessário extrair o ficheiro “repos.tgz” basta correr o comando:

```
root@uvm038:~/backup# tar xvf repos.tgz
```

Em que “x” representa “extract”, “v” de verbose e “f” de file.