

Relatório de ASIST

Sprint 2

Turma 3DL _ Grupo 68

1190353 _ Ana Barreiro

1201534 _ André Santos

1201835 _ Roberto Nogueira

1201544 _ Rui Dias

1170878 _ Milene Farias

dezembro 2022

Conteúdo

Conteúdo.....	2
Índice de figuras.....	2
USER STORY 1 – Rui.....	3
USER STORY 2 – André.....	6
USER STORY 3 – Roberto.....	8
USER STORY 4 – Milene.....	10
USER STORY 5 – Ana	11

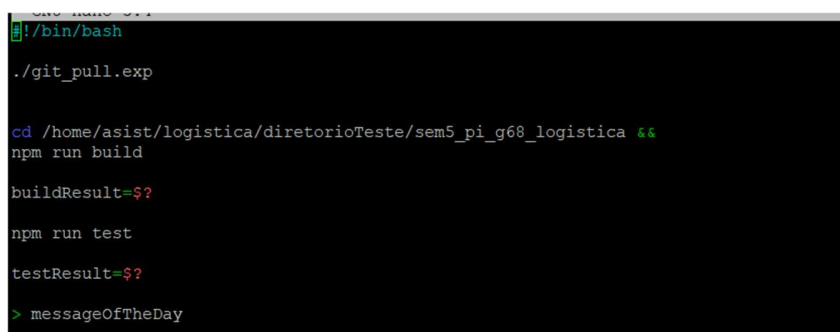
Índice de figuras

Figura 1 – Excerto 1 do ficheiro pipeline /home/asist/logística/main.sh.....	3
Figura 2 - Excerto 2 do ficheiro pipeline /home/asist/logística/main.sh.....	4
Figura 3 - Excerto do ficheiro pipeline /home/asist/logística/main.sh	4
Figura 4 - Ficheiro de configuração do Crontab.....	5
Figura 5 - logístic_deployment.service	5
Figura 6 - Comandos iptables.....	6
Figura 7 - Teste de sucesso	6
Figura 8 - Teste de falha.....	7
Figura 9 - Ficheiro de texto constituído pelas regras.....	8
Figura 10 - Script para interpretar ficheiro de texto com as regras	8
Figura 11 - Linha do ficheiro do serviço Crontab	9

USER STORY 1 – Rui

Nesta *User story* foi pedido o *deployment* sistemático de um dos módulos do projeto. Para tal foi escolhido o módulo de logística da empresa. Quanto ao *deployment*, o mesmo é executado todos os dias às 23h55 min, uma vez que é uma hora com pouco tráfego.

Para a solução desenvolvida foi feito um *script* de *pipeline* simples que dá *pull*, faz a *build*, corre testes e se não houver erros nestes últimos passos, a produção é parada e substituída pelo novo programa onde é posto a correr novamente.



```
#!/bin/bash

./git_pull.exp

cd /home/asist/logistica/diretorioTeste/sem5_pi_g68_logistica &&
npm run build
buildResult=$?

npm run test
testResult=$?

> messageOfTheDay
```

Figura 1 – Excerto 1 do ficheiro pipeline /home/asist/logística/main.sh

Como é possível observar na *Figura 1*, o *script pipeline* começa por invocar o *script* “git_pull.exp” que é responsável por dar *pull* do módulo de logística para a pasta /home/asist/logística/diretorioTeste.

Após o *pull*, é feito a *build* através do comando “npm run build” e é corrido os testes através do comando “npm run test”. Após a execução destes comandos é guardado o seu valor de retorno.

```

cd ..
cd ..

echo "*****" >> ResultOfDeployment
echo "Pipeline Began !" >> ResultOfDeployment
echo $(date) >> ResultOfDeployment

echo "Last Deployment INFO" >> messageOfTheDay
echo $(date) >> messageOfTheDay
if [[ $buildResult -eq 0 ]]
then
    echo "Build was successful" >> ResultOfDeployment
    echo "Build was successful" >> messageOfTheDay
else
    echo "Build failed!" >> ResultOfDeployment
    echo "Build failed" >> messageOfTheDay
fi

if [[ $testResult -eq 0 ]]
then
    echo "All tests passed" >> ResultOfDeployment
    echo "All tests passed" >> messageOfTheDay
else
    echo "Not all tests have passed" >> ResultOfDeployment
    echo "Not all tests have passed" >> messageOfTheDay
fi

```

Figura 2 - Excerto 2 do ficheiro pipeline /home/asist/logística/main.sh

À medida que o *script* é corrido, o sucesso ou insucesso é registado no *ResultOfDeployment* que atuará como um *log*. Ao mesmo tempo será escrito para outro ficheiro "*messageOfTheDay*", que conterá o resultado do *deployment* mais recente.

```

if [ $testResult -eq 0 ] && [ $buildResult -eq 0 ]
then
    systemctl stop logistica_deployment.service
    cp -r /home/asist/logistica/diretorioTeste/sem5_pi_g68_logistica /home/asist/logistica/producao
    systemctl start logistica_deployment.service
    echo "The code was sent to production" >> ResultOfDeployment
    echo "The code was sent to production" >> messageOfTheDay
else
    echo "The code was not sent to production" >> ResultOfDeployment
    echo "The code was not sent to production" >> messageOfTheDay
fi

echo "Pipeline Finished!" >> ResultOfDeployment

```

Figura 3 - Excerto do ficheiro pipeline /home/asist/logística/main.sh

Caso a *build* tenham sido feita com sucesso e os testes todos passarem é parado o programa e é substituído o *software* antigo pelo novo. O *software* em execução está contido no diretório produção.

Para o agendamento da execução do *script* anterior foi usado o serviço *Crontab*. Para agendar a execução foi adicionada a linha destacada na *figura 4* no ficheiro de configuração do *Crontab*.

```

GNU nano 5.4 /tmp/
# 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
# 49 12 * * 0 tar -zc /home_backup/backup.tar /home
# 45 11 * * * tar -vc /home --file=/home_backup --listed-incremental=/home_backup
55 23 * * * /home/asist/logistica/main.sh
# * * * * /root/scripts/updateTables.sh

```

Figura 4 - Ficheiro de configuração do Crontab

Para gerenciar a execução do módulo logística foi usado o *systemd*. Foi então criado o serviço presente na figura 5.

```

GNU nano 5.4
[Unit]
Description=Logistics Node Server
Documentation=https://example.com
After=network.target

[Service]
Environment=NODE_PORT=3000
Type=simple
User=asist
WorkingDirectory=/home/asist/logistica/producao/sem5_pi_g68_logistica
ExecStart=npn start
Restart=on-failure

[Install]
WantedBy=multi-user.target

```

Figura 5 - *logistic_deployment.service*

O serviço está configurado para correr na *port* 3000 no diretório produção. Foi ainda especificado o comando de execução e configurado de forma a que em caso de falha execute novamente.

USER STORY 2 – André

Este User Story tem como objetivo bloquear todos os utilizadores fora da rede interna do DEI/VPN e não permitir que estes acedam à solução. Esta regra já existe quando se pretende aceder à máquina virtual onde está a solução, isto é, não é possível aceder à VM Linux sem usar VPN do DEI. O que vai ser feito é, de certa forma, duplicar a segurança existente, usando as *iptables* do Linux.

Os IPs a serem permitidos serão da do tipo **10.8.0.0/16** que são os usados pela rede local do DEI e VPN do DEI. Todos os outros acessos serão bloqueados. As regras apenas vão bloquear ou permitir acesso à solução (módulo de logística) à qual foi feita o deployment na US1, que está a correr num serviço *systemd* e a receber pedidos na porta 3000.

Iptables no *Linux* permitem fazer isso mesmo, para o endereço acima será acrescentada uma regra de ACCEPT na porta 3000 à chain de INPUT. No final da tabela de INPUT será acrescentada uma regra de DROP na porta 3000 que faz match com todos os outros endereços (0.0.0.0/0). Por fim, as *iptables* são guardadas e podemos testar que temos acesso à solução. Teremos então estes comandos:

```
-A INPUT -s 10.8.0.1/16 -p tcp -m tcp --dport 3000 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 3000 -j DROP
```

Figura 6 - Comandos *iptables*

Os 4 endereços mencionados acima correspondem aos endereços e mascaras que vemos na imagem. De forma a testar que a implementação está correta, basta retirar as regras 1 a 4 e verificar que perdemos o acesso à solução.

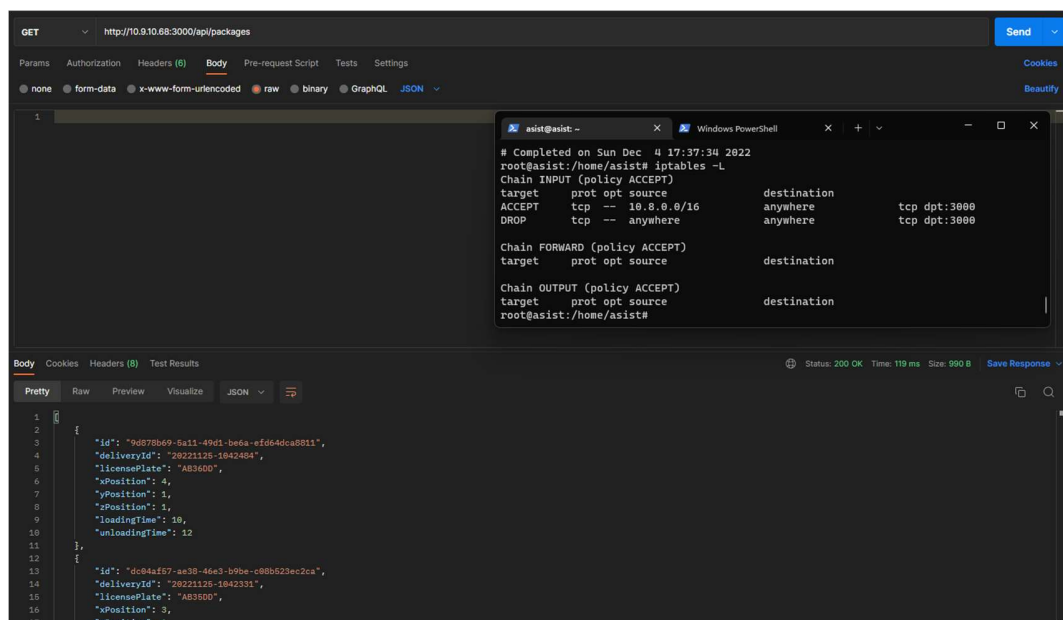


Figura 7 - Teste de sucesso

Na imagem acima vemos que quando implementadas todas as regras mencionadas anteriormente, é possível fazer um pedido à solução e receber uma resposta com código 200 OK.

Retirando a primeira regra, todos os pedidos vão deixar de ser aceites pelas iptables. Como vemos na imagem seguinte, deixa de ser possível realizar o pedido, sendo atingido o request timeout.

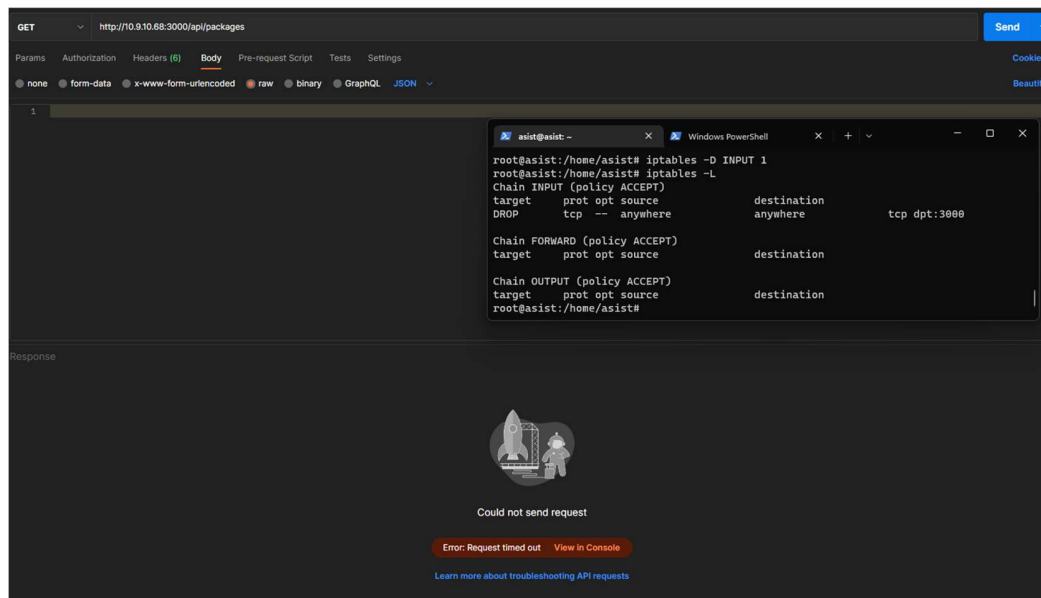


Figura 8 - Teste de falha

USER STORY 3 – Roberto

O objetivo deste *User Story* é o complemento da *User Story* anterior, ser apenas permitido o acesso aos clientes da rede interna do Departamento de Engenharia Informática para aceder à solução, contudo de uma maneira diferente, pela alteração de um ficheiro de um texto.

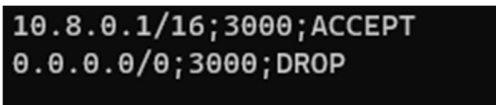
Esta *User Story* pode ser dividida em 3 passos:

- Criação de ficheiro de texto constituído pelas regras a ser adicionadas
- Criação de *script* para interpretar ficheiro de texto e criar regras
- Atualizar ficheiro do serviço *Crontab* para que o *script* seja executado periodicamente

Inicialmente, foi criado um ficheiro de texto que pode ser gerido pelo administrador do sistema. Este ficheiro irá conter várias linhas que irão dar origem às regras a ser adicionadas à *iptables*. Cada linha é uma regra e é dividida em três argumentos por um ponto e vírgula, sendo o primeiro o *source*, o segundo a porta e o terceiro o *target*. O administrador de sistema necessita de seguir o esquema abaixo quando pretender adicionar uma regra nova ao ficheiro de texto.

SOURCE;PORT;TARGET

As regras utilizadas neste ficheiro foram explicadas na *User Story* anterior.



```
10.8.0.1/16;3000;ACCEPT
0.0.0.0/0;3000;DROP
```

Figura 9 - Ficheiro de texto constituído pelas regras

Depois de feito o ficheiro de texto que contém as regras, foi feito o *script* que interpretará o ficheiro e irá adicionar as regras à *iptables*. O *script* pode ser visto na figura abaixo.



```
#!/bin/bash

iptables -F

filename='rules.txt'

while read line; do
    IFS=';' read -r -a ARRAY <<< "$line"
    if [ "${ARRAY[2]}" = "DROP" ]
    then
        iptables -A INPUT -p tcp -s ${ARRAY[0]} --dport ${ARRAY[1]} -j ${ARRAY[2]}
    else
        iptables -I INPUT -p tcp -s ${ARRAY[0]} --dport ${ARRAY[1]} -j ${ARRAY[2]}
    fi
done < $filename

iptables-save
```

Figura 10 - Script para interpretar ficheiro de texto com as regras

Inicialmente, o *script* irá eliminar todas as regras presentes na *iptables* através do comando “iptables -F”, uma vez que o *script* que irá executar periodicamente irá atualizar as regras da tabela a partir do ficheiro

do texto, que irá conter sempre as regras atualizadas. Caso a tabela não seja eliminada, poderia acontecer a mesma regra ser duplicada várias vezes ao longo da execução periódica do *script*. Para evitar isto, é sempre eliminado todos os registos da *iptables*.

Depois de eliminadas todas as regras, o ficheiro com as regras será lido linha a linha e cada linha será dividida por ponto e vírgula, para que seja avaliado se a regra tem como *target accept* ou *target drop*. Caso o *target* seja *drop*, a regra adicionada terá de ser adicionada no fim da tabela, para que primeiro possam ser aceites todos as outras regras que contenham o *target accept*. A diferença é visível no primeiro argumento do comando de adicionar uma nova regra. Caso seja “-A”, esta regra é adicionada no fim da tabela, para o caso de *drop*, caso seja “-I”, esta regra é adicionada no início da tabela.

Por fim, foi atualizado o ficheiro do serviço *Crontab* para que o *script* seja executado a cada minuto, de maneira que a tabela seja atualizada a cada minuto. Para isso, foi adicionada uma nova linha ao ficheiro, que pode ser visível na figura abaixo.

```
* * * * * /home/asist/addRules.sh
```

Figura 11 - Linha do ficheiro do serviço *Crontab*

USER STORY 4 – Milene

Como administrador quero identificar e quantificar os riscos envolvidos na solução preconizada.

Os riscos envolvidos no uso das VM são os seguintes:

1º - utiliza muitos recursos do sistema;

2º - virtualiza todo o computador;

3º - são “pesadas” para o sistema;

4º - performasse limitada;

5º - pode existir o risco de interrupção dos serviços, por sobrecarga do sistema ou problemas de hardware.

USER STORY 5 – Ana

“5 - Como administrador do sistema quero que seja definido o MBCO (*Minimum Business Continuity Objective*) a propor aos *stakeholders*”

Pretende-se definir um número mínimo de operações a serem mantidas durante uma disrupção na infraestrutura.

Períodos curtos de indisponibilidade de menos de 1 hora são aceitáveis, dadas as especificações do cliente, tendo em conta que pretendem que o sistema funcione o maior tempo possível. O sistema deve suportar o funcionamento parcial (apenas alguns módulos disponíveis).

Base de dados dos Armazéns – Nível 0

Este servidor contém toda a informação sobre as entregas que fazem no armazém, ou seja, não deve estar em baixo. Em relação ao MTD (*Maximum Tolerable Downtime* – tempo máximo de inatividade) e ao MTPD (*Maximum Tolerable Period of Disruption* – tempo máximo em que os programas principais da infraestrutura ficam indisponíveis) não devem existir.

Base de dados da Logística - Nível 0

Este servidor contém todos os dados dos camiões. Tal como no primeiro caso este serviço é fulcral, sendo da mesma forma inexistentes os MTD e MTPD.

Módulo de Armazéns - Nível 1

Este módulo serve para criar, listar, editar e eliminar armazéns e entregas. Como não há perdas de dados, não é necessário o módulo SPA (*front end*).

Em relação a MTD sugerimos que seja baixo, por exemplo 15 minutos e um MTPD baixo também.

Módulo de Logística - Nível 1

Este módulo serve para criar, listar, editar e eliminar camiões, os empacotamentos dos mesmos e as suas rotas.

Como é um serviço que depende de outro (Módulo de Armazéns e Entregas e base de dados), definimos o MTD para 15 minutos, mas o MTPD para 20 minutos.

Módulo de Planeamento - Nível 2

Este módulo serve para gerar todas as rotas e todos os armazéns disponíveis para entregas.

Como depende de outro serviço (Módulo de Logística) sugerimos um MTD superior, por exemplo 30 minutos e um MTPD de 35 minutos.

Módulo de SPA - Nível 3

Este módulo *front-end* necessita de todos os outros módulos, ou seja, não é fulcral. Definimos um MTD mais alto um pouco, por exemplo 50 minutos e o MTPD de 55 minutos.

Tendo em consideração os módulos e o tempo de indisponibilidade inferior a 1 hora, conseguimos definir um MCBO as especificações do cliente.