Sprint 3

Administração de Sistemas

Turma 3DG _ Grupo 41
1190679 – Ivo Oliveira
1190384 – André Teixeira
1201199 – Miguel Macedo
1201496 – Mário Cardoso
1201197- João Figueiredo

Data: 08/01/2023

User Story 1

O plano de recuperação de desastre é um documento que detalha os passos a serem realizados para restaurar os sistemas e processos da empresa após uma interrupção ou crise. É um componente importante do plano de continuidade dos negócios e deve ser criado antes que qualquer crise ocorra, para garantir que a empresa esteja preparada para qualquer eventualidade.
Para criar um plano de recuperação de desastre que satisfaça o MBCO da sua empresa, devemos seguir os seguintes passos:

1. Identificar os sistemas e processos críticos para a continuidade dos negócios;

2. Determinar o tempo máximo que cada um desses sistemas e processos pode ficar inativo antes que o MBCO seja violado;

3. Desenvolver um plano para restaurar cada um desses sistemas e processos o mais rapidamente possível após uma interrupção ou crise;

4. Testar o plano de recuperação de desastre periodicamente para garantir que ele esteja atualizado e funcione corretamente;

5. Atualizar o plano de recuperação de desastre conforme necessário à medida que a empresa cresce e muda;

User Story 2

Para assegurar um MTD de 20 minutos, é importante ter uma infraestrutura que seja tolerante a falhas e capaz de se recuperar rapidamente. Algumas alterações que podemos considerar são:

1. Implementar redundância em vários níveis: isto pode incluir redundância de hardware, de software e de rede. Por exemplo, pode ter servidores em cluster para garantir a disponibilidade do serviço, mesmo em caso de falha de um servidor individual.

2. Implementar soluções de backup e recuperação: é importante ter planos de backup e recuperação robustos para garantir que, em caso de falha, é possível recuperar rapidamente os dados e os sistemas críticos.

3. Monitorizar a infraestrutura em tempo real: a monitorização em tempo real permite detetar e corrigir problemas rapidamente, minimizando o tempo de inatividade.

4. Realizar manutenção preventiva regularmente: a manutenção preventiva pode ajudar a evitar falhas na infraestrutura e garantir o bom funcionamento.

5. Ter um plano de ação em caso de falha: é importante ter um plano de ação detalhado para lidar com falhas de forma rápida e eficaz. Este plano deve incluir a lista de procedimentos responsáveis por cada tarefa e os m utilizar.

User Story 3

Para realizar uma cópia de segurança do MongoDB para um ambiente de nuvem através de um script, podemos usar os comandos mongodump e mongorestore. O comando mongodump pode ser usado para criar um ficheiro de backup da base de dados MongoDB, enquanto o comando mongorestore pode ser usado para restaurar a base de dados a partir do arquivo de backup.
De seguida apresentamos um exemplo de como criar um script que fará uma cópia de segurança da base de dados MongoDB e a renomeará para o formato <nome_da_db>_yyyymmdd:

Primeiro obtemos o nome da base de dados que queremos fazer backup:

```
echo "Digite o nome da base de dados que deseja fazer backup:"
read db_name
```

Depois obtemos a data atual:

```
date=$(date +%Y%m%d)
```

Criamos o arquivo de backup através do comando mongodump:

```
mongodump --db $db_name --out /caminho/para/arquivos/de/backup
```

Renomeamos o arquivo de backup para o formato <nome_da_db>_yyyymmdd:

```
mv /caminho/para/arquivos/de/backup/$db_name
/caminho/para/arquivos/de/backup/${db_name}_${date}
```

Enviámos o arquivo de backup para o ambiente de nuvem através do comando scp:

```
scp /caminho/para/arquivos/de/backup/${db_name}_${date}
```

Para realizar uma cópia de segurança do MySQL e enviá-la para um ambiente de nuvem, podemos usar o comando **mysqldump**. Este comando cria um arquivo de dump que contém uma representação em SQL da base de dados.

Para incluir a data no nome do arquivo dump, usamos o comando **date** do Linux. Por exemplo:

```
mysqldump -u username -p database_name > database_name_$(date +%Y%m%d).sql
```

Este comando cria um arquivo de dump com o nome **database_name_yyyymmdd.sql**, onde **yyyy** é o ano atual, **mm** é o mês atual e **dd** é o dia atual.

Para enviar o arquivo de dump para o ambiente de nuvem, usamos o comando **scp** (secure copy). Por exemplo:

```
scp database_name_$(date +%Y%m%d).sql username@remote_server:/remote/path
```

Este comando envia o arquivo de dump para o servidor remoto na pasta especificada.

Se quisermos automatizar este processo, podemos criar um script que execute os comandos acima. Por exemplo, criamos um script chamado **backup_mysql.sh** com o seguinte conteúdo:

```
mysqldump -u username -p database_name > database_name_$(date +%Y%m%d).sql
scp database_name_$(date +%Y%m%d).sql username@remote_server:/remote/path
```

Depois para executar o script, basta dar permissão de execução e chamá-lo:

```
chmod +x backup_mysql.sh
./backup_mysql.sh
```

User Story 4

Para criar um script que gere os arquivos de backup resultantes do script de backup feito na alínea anterior, podemos usar os comandos find e rm do Linux. O comando find pode ser usado para procurar por arquivos com base em vários critérios, enquanto o comando rm pode ser usado para excluir arquivos.

De seguida apresentamos um exemplo de como criar um script que fará a gestão dos arquivos de backup resultantes do script de backup anterior, de acordo com o calendário descrito:

Primeiro definimos o caminho para a pasta de arquivos de backup:

```
backup_dir="/caminho/para/arquivos/de/backup"
```

Depois obtemos a data atual:

```
date=$(date +%Y%m%d)
```

Obtemos o último dia do mês anterior:

```
last_month=$(date -d "$(date +%Y-%m-01) -1 day" +%Y%m%d)
```

Fazemos o mesmo, mas para o último dia da semana anterior:

```
last_week=$(date -d "$(date +%Y-%m-%d) -7 days" +%Y%m%d)
```

Utilizamos o comando find para excluir os arquivos de backup mais antigos que o último mês:

```
find $backup_dir -type f -name "*.gz" -mtime +30 -delete
```

Utilizamos o mesmo recurso, mas para excluir os arquivos de backup mais antigos que a última semana, mas mais novos que o último mês

```
find $backup_dir -type f -name "*.gz" -mtime +7 -mtime -30 -delete
```

Por último, excluímos os arquivos de backup mais antigos que o último dia, mas mais novos que a última semana

```
find $backup_dir -type f -name "*.gz" -mtime +1 -mtime -7 -delete
```

User Story 5

Para manter um log do processo de cópia de segurança do MongoDB e ser alertado em caso de falha, devemos fazer o seguinte:

1. Adicionar os comandos do processo de cópia de segurança a um arquivo de script shell, como descrito anteriormente.

```
mongodump --db <nome_da_db> --out /caminho/para/diretorio/de/backup
```

2. Adicionar o comando **set -e** no início do script para fazer com que o script saia em caso de erro.

```
set -e
```

3. Adicionar o comando **set -o pipefail** logo abaixo do comando **set -e** para fazer com que o script saia em caso de erro ao usar um pipeline.

```
set -o pipefail
```

4. Adicionar o comando **exec &>> /caminho/para/arquivo/de/log/backup.log** no final do script para redirecionar a saída do script para um arquivo de log. Isso permitirá que mantenhamos um registo do processo de cópia de segurança.

```
exec &>> /caminho/para/arquivo/de/log/backup.log
```

5. Adicionar um **trap** no final do script para capturar eventos de erro e enviar uma notificação ao administrador

```
trap 'echo "Falha na cópia de segurança. Verifique o arquivo de log para obter
mais informações." | mail -s "Falha na cópia de segurança"
administrador@example.com' ERR
```

User Story 6

Para garantir que as cópias de segurança anteriores têm um tempo de segurança inferior a 7 dias, uma possível solução é adicionar uma etapa ao script que remova as cópias mais antigas.

Remover cópias de segurança do MongoDB:

```
find /db_backups/mongodb -name "mongodb_*" -mtime +7 -delete
```

Remover cópias de segurança do MySQL:

```
find /db_backups/mysql -name "mysql_*" -mtime +7 -delete
```

Para isso, podemos utilizar o comando find do que vai procurar arquivos no caminho especificado e cujo o nome comece com “db_name” e que tenham sido alteradas há mais de 7 dias (-mtime +7). Podemos utilizar também o comando “-delete” para remover esses arquivos encontrados.

User Story 7

Um BIA é um processo usado para avaliar os impactos de uma interrupção. Ajuda a identificar os recursos críticos da empresa e avaliar qual seria o impacto de uma interrupção nesses recursos.

Para criar um BIA da solução final, podemos seguir os seguintes passos:

1. Identificar os recursos críticos da empresa. Podem incluir equipamentos, sistemas de informação, fornecedores, funcionários, etc;

2. Analisar o impacto de uma interrupção nos recursos críticos. Considere o impacto financeiro, o impacto no tempo de inatividade e qualquer outro impacto que uma interrupção possa ter;

3. Determinar o tempo de recuperação aceitável para cada recurso crítico. Isso é o tempo que a empresa pode tolerar sem o uso do recurso antes que isso tenha um impacto significativo nos negócios;
4. Adaptar o BIA para incluir os riscos identificados na solução. Considerar como os riscos podem afetar os recursos críticos e como eles podem ser gerenciados;

5. Monitorizar continuamente o BIA. É importante rever o BIA periodicamente para garantir que ainda está preciso e relevante e para identificar quaisquer mudanças nos riscos ou nos recursos críticos da empresa;

User Story 8

Uma gestão de acessos eficiente é importante para garantir a segurança da sua organização. Existem algumas medidas que podem ser tomadas para implementar uma gestão de acessos adequada:

1. Definir políticas de acesso: é importante definir regras para o uso dos sistemas e dados da sua organização. Isso inclui coisas como quem tem permissão para aceder aos sistemas e dados, como os utilizadores, devem autenticar-se para obter acesso e como gerenciar senhas;

2. Implementar autenticação de dois fatores: a autenticação de dois fatores adiciona uma camada extra de segurança, exigindo que os utilizadores forneçam algo que eles sabem (como uma senha) e algo que eles têm (por exemplo um código gerado pelo telefone). Isso ajuda a evitar que invasores consigam aceder aos sistemas da sua organização com senhas roubadas;

3. Monitorizar o acesso: é importante monitorizar quem está a aceder aos sistemas e dados da sua organização e quais as ações que estão a ser realizadas. Isto pode ajudar a detetar e prevenir atividades maliciosas;

4. Atualizar as senhas regularmente: as senhas fracas são uma das principais vulnerabilidades de segurança. É importante exigir que os utilizadores atualizem as senhas regularmente e usem senhas fortes;

5. Treinar os funcionários: é importante que todos os funcionários da sua organização estejam cientes das políticas de segurança e saibam como proteger os sistemas e dados da empresa. Ter formações regularmente pode ajudar a garantir que todos estejam cientes das melhores práticas de segurança;

User Story 9

Clustering é um processo que consiste em agrupar objetos (no caso, sistemas) em conjuntos (clusters) de forma que os objetos num mesmo cluster sejam mais similares entre si do que aos objetos de outros clusters. Isso pode ser útil em várias situações, como por exemplo:

1. Para melhorar a disponibilidade do sistema, criando uma configuração em que os sistemas num mesmo cluster possam substituir-se uns aos outros em caso de falha;

2. Para balancear a carga de trabalho entre os sistemas, distribuindo as requisições entre os diferentes clusters de forma equilibrada;

3. Para facilitar a manutenção e atualização do sistema, permitindo que essas operações sejam realizadas em conjunto para um cluster de sistemas de cada vez, em vez de precisar de fazê-las individualmente para cada sistema;

4. Para permitir que o sistema escalasse horizontalmente adicionando mais sistemas aos clusters conforme a demanda aumenta.

Para justificar a implementação de um sistema de clustering, podemos ter em consideração os benefícios que traria para a organização, como os que foram mencionados anteriormente, e também avaliar se os custos (em termos de tempo, esforço e recursos) envolvidos na implementação do sistema seriam justificáveis em relação a esses benefícios. Além disso, pode ser útil fazer uma análise de viabilidade do projeto, avaliando se os sistemas atuais da organização possuem a capacidade de serem configurados em clusters e se isso é tecnicamente viável.

User Story 10

1. No computador local, geramos uma chave pública e privada utilizando o OpenSSH. Devemos certificar-nos de proteger a chave privada com uma palavra-passe. Por exemplo:

```
ssh-keygen -t rsa -b 4096 -f chave_privada
```

   Isto irá gerar as chaves **chave_privada** (chave privada) e **chave_privada.pub** (chave pública). Mais uma vez devemos certificar-nos de proteger a chave privada com uma palavra-passe forte e de nos lembrar da mesma.

2. Transferimos a chave pública para a máquina virtual utilizando o comando **ssh-copy-id** para fazer isto de forma simples. Por exemplo:

```
ssh-copy-id -i chave_privada.pub usuario@endereco_ip
```

   Isto irá adicionar a chave pública ao ficheiro **~/.ssh/authorized_keys** do utilizador **usuario** na máquina virtual.

3. Editamos o ficheiro de configuração **/etc/ssh/sshd_config** da máquina virtual e alteramos a seguinte linha:

```
# Change to no to disable tunnelled clear text passwords
PasswordAuthentication yes
```

   para:

```
# Change to no to disable tunnelled clear text passwords
PasswordAuthentication no
```

4. Reiniciamos o serviço SSH para aplicar as alterações com o comando **systemctl restart ssh**.

Agora, para aceder à máquina virtual utilizando o SSH, teremos de utilizar a chave privada e a palavra-passe que protege a mesma. Por exemplo:

```
ssh -i /caminho/para/chave_privada usuario@endereco_ip
```