1.

1.1 Quais são as 5 características essenciais que definem uma oferta de Cloud Computing? Enuncie-as apenas.

On-demand self-service / “self-service”/serviço pelo próprio, a pedido

Broad network access / acesso por tecnologias de rede amplamente suportadas

Resource pooling / agrupamento de recursos com economias de escala

Rapid elasticity / elasticidade automática e “rápida”

Measured service / serviço medido, com granularidade detalhada

1.2 Exemplifique as características que respondeu em 1.1, em ação num hipotético serviço de mensagens de “lembretes”. Neste serviço, os utilizadores registados criam “lembretes”, que são mensagens agendadas para serem enviadas automaticamente pelo serviço, em momento(s) configuráveis. Organize a sua resposta por característica.

O serviço de lembretes, que poderia ser disponibilizado num URL como “https://remember.me”, teria que suportar a autonomia dos seus administradores e utilizadores, para funções como criação/modificação/remoção de contas, e todas as operações desejadas sobre os “lembretes”. Isto é uma forma de respeitar o serviço pelos próprios, a seu pedido.

O serviço seria oferecido pelas tecnologias mais estabelecidas na Internet, sem necessidade de camadas de rede ou aplicativos extra, de forma a que qualquer dispositivo com acesso à Internet lhe possa ter acesso, tanto para administração, bem como utilização. Isto corresponde a um alinhamento com o acesso por tecnologias de rede amplamente suportadas.

As funcionalidades do serviço deveriam escalar, sem latência significativa, para baixo ou para cima, em função do nível de procura, de forma a manter a experiência de utilização sem degradação, em situações de stress; e de forma a minimizar

custos, em cenário de pouca procura – isto representa a elasticidade automática e “rápida”.

Os recursos, como base de dados e instâncias computacionais, poderiam ser alocados de um agrupamento (pool) de recursos partilhados com a generalidade dos outros clientes do Cloud Provider, para beneficiar dos custos menores decorrentes de economias de escala. Ou então, se, por exemplo, considerar-se que a informação envolvida é demasiado sensível para estar em Cloud pública, fazer um uso exclusivo de alguma pool (Cloud privada).

Tanto para administradores, como para utilizadores finais, as operações realizadas terão que ser monitorizadas e cobrada(s) de acordo com preçário transparente, o que torna o serviço, um serviço medido, com granularidade detalhada.

2. Verdadeiro (V) ou Falso (F)?

2.1 À medida que se transita de IaaS, para PaaS, para SaaS, aumenta o que se abstrai, e mais opções ficam limitadas às escolhas do Provider. **V**

2.2 No curto prazo, a opção por Cloud Computing é tendencialmente mais dispendiosa do que a opção por recursos próprios. **F**

2.3 Cloud Computing corresponde à opção por uma gestão orientada a OpEx, e não orientada a CapitalEx. **V**

2.4 Uma “Cloud Drive”, como “DropBox”, “Google Drive”, etc., é uma situação de PaaS. **F**

2.5 Se um utilizador perde ficheiros importantes numa “Cloud Drive”, porque a “Cloud Drive” falha, a responsabilidade é toda do Cloud Provider. **F**

2.6 O isolamento por deployment em contentor é inferior ao isolamento por deployment em máquina virtual. **V**

3.

Imagine-se a fazer desenvolvimento em AWS Cloud 9, numa compute instance acabada de criar; ou seja, por exemplo, o Git (e outros utilitários) está disponível, mas não configurado.

Existe ainda um repositório remoto relevante, público, de nome “repo240517”, em github.com, do utilizador “cloud24”. O URL https do repo é:

https://github.com/cloud24/repo240517

Indique os comandos git e/ou bash, necessários para:

3.1. “Clonar” o repositório para a pasta corrente

git clone https://github.com/cloud24/repo240517

3.2. Listar os branches existentes.

git branch

3.3. Assuma que só existe um branch, de nome “master”, e que está precisamente nesse contexto. Como listar os ficheiros que fazem parte do commit mais recente em master?

git ls-tree HEAD

3.4. Crie um novo ficheiro 240517.txt, sem conteúdo.

touch 240517.txt

3.5. Acrescente 240517.txt ao stage.

git add 240517.txt

3.6. É possível fazer um novo commit? Sim ou não? Justifique.

Não. Se o git não está configurado, o commit será rejeitado. Primeiro, haverá que fazer p git config adequado.

3.7. Admita que se fez um novo commit com mensagem “245017-1”, que inclui o ficheiro “240517.txt”. Crie e mude o contexto de trabalho para um novo branch de nome “new”.

git checkout -b new

3.8. Modifique “240517.txt” e faça um novo commit com mensagem “240517-2”.

echo “extra”>>240517.txt

3.9. Regresse ao ramo master.

git checkout master

3.10. Qual o conteúdo de “240517.txt” depois do comando git checkout HEAD 240517.txt ?

Nenhum: o ficheiro estava vazio de conteúdo, quando foi commited.

4. Nesta questão, o objetivo é escrever uma app **Python Flask**, a ser Cloud deployed, que saiba responder a pedidos **GET** com a seguinte estrutura:

http://<domain name>:<port>/results/<year>/<month>

por exemplo: http://localhost:5005/results/2024/5

Os pedidos terão origem numa interface HTML muito simples, ficheiro "**interface.html**", com conteúdo que poderá ler adiante, portanto, duas caixas numéricas, uma para escrever um ano, outra para escrever um mês. Esta interface deve ser servida na route /

Existe ainda um ficheiro "**interface.js**", cujos detalhes não são relevantes, nem terá que escrever, mas que assegura a invocação dos URLs certos, em função do que pede na form.

```
<html><!-- interface.html -->
...
    <form id="idForm">
        <fieldset>
            <legend>Year and month:</legend>
                <label for="idYear">Year:</label>
                <input type="number" id="idYear"
placeholder="year" value="2024"><br>
                <label for="idMonth">Month:</label>
                <input type="number" id="idMonth"
placeholder="month" value="5">
            </fieldset>
        <input type="submit" value="get the results with
year/month">
    </form>
    <hr>
    <section id="idFeedback"></section>
...
</html>
```

Assuma que esta app (**app.py**) tem acesso a uma base dados de todos os resultados alguma vez sorteados no jogo de azar "EuroMillions".
A base de dados está disponível num ficheiro de texto "**DB.JSON**", em formato JSON.
O formato do ficheiro é exatamente o formato das respostas que têm que ser produzidas, exemplificado adiante.

Os URLs http://<domain name>:<port>/results/<year>/<month> permitem obter respostas de texto, estruturadas em JSON. Cada resposta é uma estrutura JSON que representa a lista dos sorteios correspondentes ao ano e mês indicados.

Segue-se um exemplo da resposta para ano de 2020, mês 2, limitada a um só sorteio.
Como se observa, cada resposta é um dicionário com chaves **mDrawNumber**, **mYear**, **mMonth**, **mDay**, **mBalls** e **mStars**.
Para este exercício, só é importante compreender que a chave **mYear** representa o ano do sorteio; e a chave **mMonth** o mês do sorteio.

```
[
{"mDrawNumber": 1298, "mYear": 2020, "mMonth": 2, "mDay": 28, "mBalls": [8, 11, 23, 20, 22], "mStars": [4, 3]},
…
]
```

4.1. Escreva uma “árvore” de diretorias e ficheiros, que indique claramente a organização da sua app, adequada a deployment nalgum ambiente Cloud capaz de suportar os seus requisitos.

```
/app240517/
|-- app.py
|-- requirements.txt
|-- templates/
|   |-- interface.html
|   |-- search_results.html
|-- static/
    |-- DB.JSON
    |-- interface.js
```

4.2. Escreva o código Python necessário para operacionalizar a app. Não utilize técnicas de “list comprehension”.

TODO

4.3. Indique as modificações (HTML e Python) necessárias, para que a resposta passe a ser comunicada para uma página **search_results.html**, em que cada resultado aparece como membro de uma unordered list (ul).

Em app.py, no método que retorna os resultado da procura:

```
return render_template(
    'search_results.html',
    results=listWanted
)
```

em **search_results.html**:

```
<body>
    <ul id="resultsList">
        {% for result in results %}
            <li>Draw Number: {{ result.mDrawNumber }},
Date: {{ result.mYear }}-{{ result.mMonth }}-{{
result.mDay }}, Balls: {{ result.mBalls }}, Stars: {{
result.mStars }}</li>
        {% endfor %}
    </ul>
</body>
```