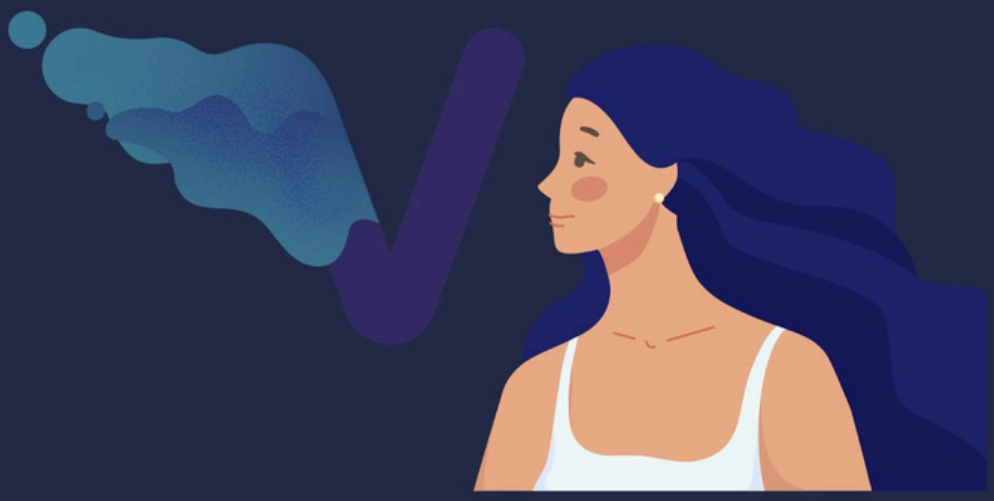
VALCANN
CLOUD INTELLIGENCE
An EPI-USE company

OPORTUNIDADE

PROGRAMA DE ESTÁGIO



Programa de Estágio 2025.1

Avaliação Técnica

Sumário

Visão geral da Avaliação Técnica

Olá!
É um prazer poder contar com a sua participação em nosso Programa de Estágio. O objetivo deste documento é apresentar 3 (três) casos de negócios, os quais deverão ser analisados e a partir desta análise, serem propostas soluções práticas para os problemas. Aqui vão algumas orientações:

1. Todas as suas respostas deverão ser adicionadas neste documento, em suas respectivas seções;
2. Para os casos apresentados, não existe apenas uma resposta certa. Orientamos você a responder com suas palavras e seu conhecimento, evitando reproduzir "respostas ideais" ou que possam caracterizar a proposição desta forma. O objetivo é avaliar a sua capacidade de endereçar os problemas apresentados, com os recursos e ferramentas que possui;
3. Suas respostas deverão ser enviadas para o email faleconosco@valcann.com.br, com o título VALCANN | Programa de Estágio 2023.2 | Resposta Avaliação Técnica | SEU NOME;

4. Os problemas devem ser resolvidos individualmente por cada pessoa candidata. Durante o processo, poderemos requisitar que você apresente suas propostas nas reuniões com nosso time de recrutamento.

Problema 1 | Automação de Ambientes Operacionais

Um dos principais desafios para um bom gerenciamento de infraestrutura, é implementar automação para permitir produtividade aos times de administração de tecnologia, bem como, minimizar ações humanas nos ambientes dos clientes.

O cliente “Acme Co.” possui um servidor centralizado de backup, o qual recebe arquivos de todos os demais servidores, move os dados para um volume temporário, para que deste volume os dados sejam copiados por uma ferramenta de backup externa.

De forma a minimizar o nível de intervenção neste ambiente, você foi convocado a escrever um script (em Shell Script, Python ou qualquer outra tecnologia que preferir), para automatizar as seguintes ações:

Listar todos arquivos (nome, tamanho, data de criação, data da última modificação) localizados no caminho /home/valcann/backupsFrom;

Salvar o resultado no arquivo backupsFrom.log em /home/valcann/;

Remover todos os arquivos com data de criação superior a 3 (três) dias;

Copiar todos os arquivos os arquivos com data de criação menor ou igual a 3 (três) dias em /*home*/valcann/backupsTo;

Salvar o resultado no arquivo backupsTo.log em /home/valcann/.

Fique à vontade para pesquisar ou reutilizar códigos disponíveis na comunidade.

R. De acordo com as necessidades informadas pela Acme Co. relacionadas ao processo de automação de backup dos documentos da empresa, foi escrito o código abaixo em python, para realização dos processos solicitados:

```
import os
import shutil
from datetime import datetime, timedelta

 def get_file_info(filepath):

   status = os.stat(filepath)
   created = datetime.fromtimestamp(status.st_ctime)
   modified = datetime.fromtimestamp(status.st_mtime)
   size = status.st_size
   return {
       'name': os.path.basename(filepath),
       'size': size,
       'created': created,
       'modified': modified
   }

 def format_file_info(fileinfo):

   return(f"Nome: {fileinfo['name']}, "
       f"Tamanho: {fileinfo['size']} bytes, "
       f"Creation Date: {fileinfo['created'].strftime('%Y-%m-%d %H:%M:%S')}, "
       f"Modification Date: {fileinfo['modified'].strftime('%Y-%m-%d %H:%M:%S')}")

 def main():

   source_dir = "/home/valcann/backupsFrom"
   destination_dir = "/home/valcann/backupsTo"
   log_dir = "/home/valcann"

   for directory in [source_dir, destination_dir]:
       if not os.path.exists(directory):
           os.makedirs(directory)

   limitDate = datetime.now() - timedelta(days=3)

   with open(os.path.join(log_dir, "backupsFrom.log"), "w") as log_file:
       log_file.write(f"Log created in: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n\n")
       for filename in os.listdir(source_dir):
           filepath = os.path.join(source_dir, filename)
           if os.path.isfile(filepath):
               fileinfo = get_file_info(filepath)
               log_file.write(f"{format_file_info(fileinfo)}\n")

   copied = []
   for filename in os.listdir(source_dir):
       filepath = os.path.join(source_dir, filename)
       if os.path.isfile(filepath):
           fileinfo = get_file_info(filepath)

       if fileinfo['created'] < limitDate:
           os.remove(filepath)
       else:
           destination_path = os.path.join(destination_dir, filename)
           shutil.copy2(filepath, destination_path)
           copied.append(fileinfo)

   with open(os.path.join(log_dir, "backupsTo.log"), "w") as log_file:
       log_file.write(f"Log created in: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}\n\n")
       log_file.write(f"Total files copied: {len(copied)}\n\n")
       for fileinfo in copied:
           log_file.write(f"{format_file_info(fileinfo)}\n")

if __name__ == "__main__":
   try:
       main()
       print("Backup completed successfully!")
   except Exception as e:
       print(f"Backup Failed: {str(e)}")
```

Problema 2 | Monitoramento e Performance

Um dado cliente tem se queixado continuamente de lentidão em uma das suas principais aplicações. Trata-se de uma aplicação web, com 4 (quatro) servidores de aplicação, 2 (dois) servidores de banco de dados em replicação – sendo um nó de escrita e outro de leitura.

Ao analisar os indicadores de monitoramento de infraestrutura, identificou-se que nem a aplicação nem o banco de dados apresentaram nenhum sinal de sobrecarga ou pico de utilização de recursos. Ainda assim, a aplicação encontra-se lenta para os usuários.

Descreva em detalhes quais seriam as ações e ferramentas que você aplicaria para diagnosticar e consequentemente corrigir o problema de performance apresentado pelo cliente. É importante que você descreva com o máximo de detalhes, orientando a análise sob uma perspectiva de Problema > Causa > Solução.

Sua solução deverá incluir um diagrama de arquitetura.

R.: Ao realizar a análise da aplicação do cliente, conforme informações passadas, iremos partir dos seguintes pontos:

1 – Problema: Lentidão do acesso dos usuários a aplicação.

2 – Possíveis Causas:

2.1 – Camada de Rede:

- Latência entre o usuário e a aplicação.
- Problemas com DNS.
- Configurações do Load Balancer.
- Problemas de Rede entre os servidores.

2.2 – Camada de Aplicação:

- Problemas de Cache.
- Problemas de conexão com os Bancos de Dados.

2.3 – Camada de Rede:

- Delay na Replicação.
- Problemas de Indexação.
- Problemas na configuração da pool de conexões.

3 – Ferramentas de Diagnóstico

3.1 – Camada de Usuário:

- Chrome DevTools / HTTP Toolkit: Tempo de carregamento de recursos, performance do Javascript no lado do cliente, FCP e LCP.

3.2 – Camada de Rede:

- Ping e traceroute: verificar latência.
- Tcpdump e Wireshark: verificação e análise dos pacotes.
- MTR : Análise de Rotas.
- Logs do Load Balancer: Verificar possíveis falhas.

3.3 – Camada de Aplicação:

- Monitoramento de Logs, rastreamento de requisições e visualização de serviços.

3.4 – Camada de Banco de Dados:

- Verificação de Logs, Métricas, Slow Query, Monitoramento de Replicação e análise de índices e estatísticas.

4 – Soluções:

Caso o problema ser na Camada de Rede:

- Otimizar as configurações de DNS.
- Ajustar timeouts no Load Balancer.
- Implementar CDN (Content Delivery Network).

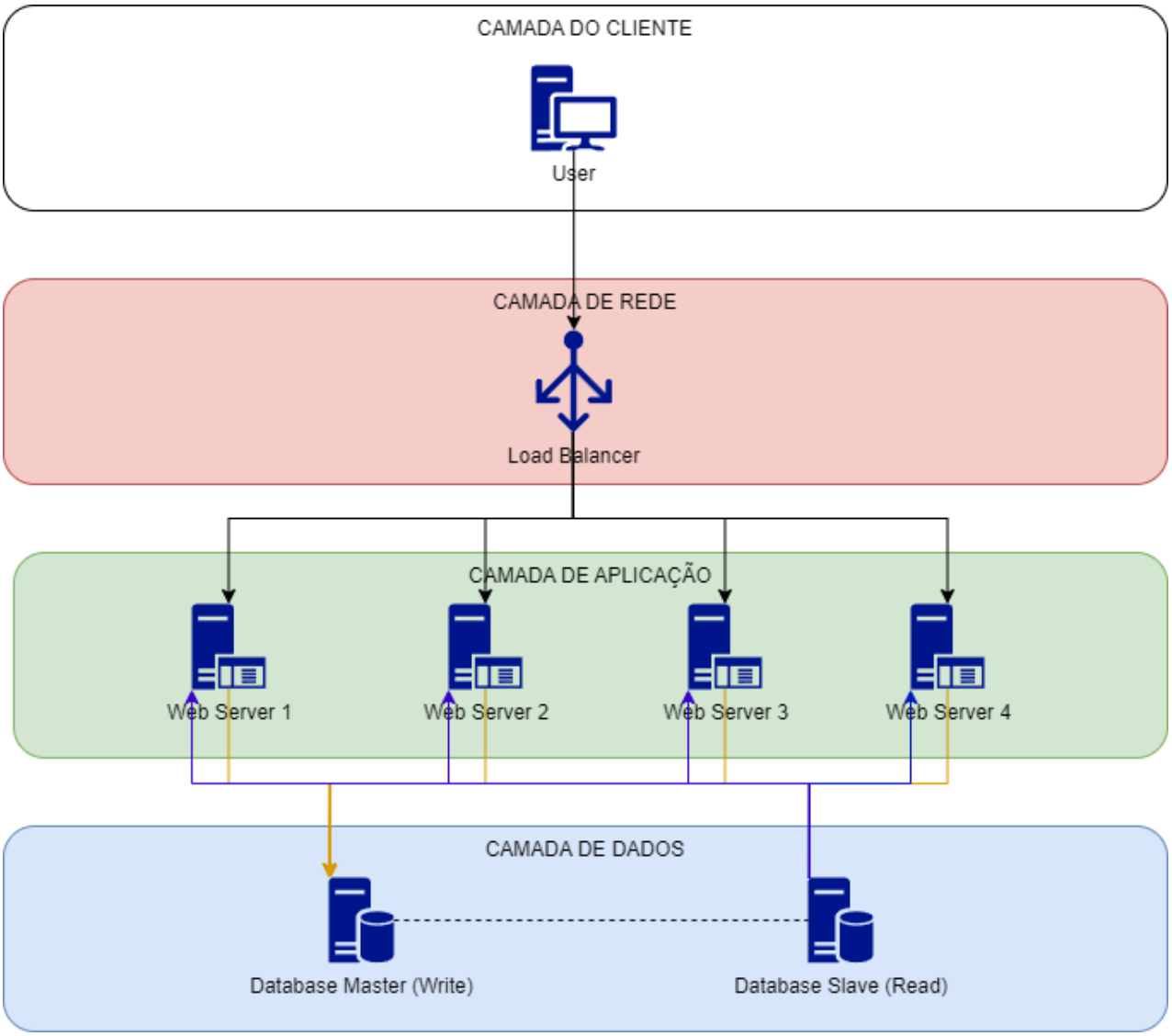
Caso o problema seja na Camada de Aplicação:

- Implementar/Otimizar Cache.
- Ajustes na pool de conexões.
- Correção de Memory Leaks.
- Otimização de código em operações mais lentas.

Caso o problema seja na Camada de Banco de Dados:

- Otimização de Queries.
- Ajustar os Índices.
- Configurar de maneira correta a replicação.

O Diagrama a seguir exemplifica a Infraestrutura da Aplicação:



Problema 3 | Aplicações e Desenvolvimento de Software

Uma empresa possui um software desenvolvido com tecnologia Node.JS no backend e React no front end. Atualmente, sempre que uma nova versão do sofware é lançada, o cliente precisa empacotar todos os componentes de front end, todos os componentes de back end e manualmente realizar o deploy no ambiente de homologação. Após uma semana de validações, há um novo empacotamento e então o ambiente de produção é atualizado.

Descreva em detalhes quais seriam as ações e ferramentas que você aplicaria para automatizar esse processo tanto na fase de homologação, quanto na fase de produção. É importante que você descreva com o máximo de detalhes, orientando a análise sob uma perspectiva de Problema > Causa > Solução.

Sua solução deverá incluir um diagrama de arquitetura.

R.: De acordo com o que foi informado pelo usuário, iremos analisar os seguintes pontos referentes a solicitação:

1 – Problema: Versionamento da Aplicação feita de forma Manual.

2 – Causa: Falta de uma Pipeline CI/CD, fazendo que os processos tenham quer ter uma intervenção manual para realização de varias partes do processo. Sua consequência é que cada deploy da aplicação deve ser realizada manualmente.

3 – Solução: Criação de uma Pipeline CI/CD com as seguintes implementações:

3.1 – Controle de Versão (SCM – Source Code Management): Usaremos um Repositório Git (Github, Gitlab, etc.) implementando o Git Flow como metodologia de criação de branches, tendo branches específicas para features, releases e hotfixes.

3.2 – Automação: Utilização de ferramentas de automação que serão responsáveis por:

- Executar testes automaticamente (testes unitários e de integração).
- Análise de qualidade de código.
- Construção para imagens Docker para o front-end e o back-end.
- Publicar as imagens em um Container Registry (Docker Hub, Amazon ECR, Etc.).
- Execução automática de deploys.

3.3 – Containerização: Utilização do Docker para criação das imagens, como também:

- Criação de Dockerfiles otimizados para cada componente da aplicação.
- Implementação de Compilação de Vários Estágios (multi-stage builds) para redução do tamanho final das imagens.
- Docker Compose para gerenciamento local dos serviços.

3.4 – Homologação: Nessa fase iremos implementar:

- Deploy automático da versão após sucesso nos testes.
- Orquestração de containers utilizando Orquestradores.
- Implementação de testes de API e Interface automatizados.
- Notificação automática para o time de Quality Assurance (QA).
- Monitoramento contínuo de métricas de performance.

3.5 – Produção: Será implementado:

- Deploy automático após aprovação manual.
- Utilização de deploy Blue/Green para downtime zero.
- Rollback automático caso ocorra falhas.
- Monitoramento avançado com alertas.

3.6 – Monitoramento: Nessa faze será realizada:

- Verificação de Logs e Visualização de Métricas.
- APM (Application Performance Monitoring) para rastreamento.
- Personalização de Dashboards para diferentes stakeholders.

3.7 – Segurança

- Verificação de vulnerabilidades nas imagens Docker.
- Análise de dependências com SAST (Static Application Security Testing)
- Gerenciamento de Segredos.
- Políticas de segurança nos clusters.

Como forma de Demonstração do Pipeline CI/CD, Segue o Diagrama abaixo: