Documento: Resultado_Desafios_Vaga_Cloud_Engineer_Ana_Lúcia_Basto
Data - Hora: 14/06/18 - 10:45:00

---------- DESAFIO - Parte 1 ----------

Estamos enviando pelos links abaixo alguns binários executáveis (ELF 64-bit LSB) que realizam tarefas bem simples, que podem ou não ser úteis. O exercício é que você descubra o que esses binários fazem, utilizando as ferramentas que julgar mais adequadas.
Como resposta, esperamos que você nos diga o que você acha que eles fazem e quais foram as ferramentas usadas para isso, bem como uma linha geral do seu raciocínio para chegar às conclusões.

Binários:

*** Utilizei o comando "hte" para visualizar e analisar o conteúdo dos arquivos binários. Através do comando "file" identifiquei que todos possuem a descrição de "ELF 64-bit LSB Executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib86/ld-linux-x86-.so.2".

https://s3.amazonaws.com/chaordic-desafio-cloud/ddb1c9

Resposta: Pendente, pois não consegui identificar nenhuma informação relevante.

https://s3.amazonaws.com/chaordic-desafio-cloud/da87fa - Pendente, pois não consegui identificar.

Resposta: Pendente, pois não consegui identificar nenhuma informação relevante.

https://s3.amazonaws.com/chaordic-desafio-cloud/d3ea79

Resposta: Pelo que verifiquei acho que pode ser o binário do jogo "The Game of Life" ou a descrição do mesmo.

https://s3.amazonaws.com/chaordic-desafio-cloud/cc9621

Resposta: Pelo que verifiquei o binário contém chamadas para "Java Class Registration (.jcr)", mas não consegui identificar o que ele faz.

---------- DESAFIO - Parte 2 ----------

O ambiente deve ser todo configurado através de gerenciador de configuração, o que deverá ser entregue é um repositório git contendo os arquivos de configuração que serão aplicados em uma máquina virtual "zerada". Caso necessário descrever como executar o processo de aplicação da configuração na máquina virtual. Ao final da tarefa e execução do processo, deveremos ter um ambiente funcional;

É recomendado que o repositório git seja entregue com commits parciais, mostrando a evolução de seu código e pensamento. Caso prefira nos informe um url de git público ou então compacte todos os arquivos em um .tar.gz mantendo a pasta .git em conjunto.

Resposta: Os arquivos de configuração foram disponibilizados na URL de Git público "https://github.com/anabasto/appnodejs1"

No ambiente deverá estar rodando uma aplicação node.js de exemplo, conforme código abaixo. A versão do node.js deverá ser a última versão LTS disponível em: https://nodejs.org/en/download/. A aplicação node abaixo possui a dependência da biblioteca express. Garanta que seu processo de bootstrap instale essa dependência (última versão estável disponível em: http://expressjs.com/) e rode o processo node em background.

De uma forma dinâmica garanta que seja criado uma instância node para cada processador existente na máquina (a máquina poderá ter de 1 a 32 processadores)

Resposta: A aplicação node.js foi configurada, conforme as recomendações acima, utilizando o comando "npm", o módulo "pm2"(distribuição entre vários processadores e monitoração da aplicação) e o comando "systemctl" (garantia da ativação da aplicação
após a reinicialização do servidor).

----- Código da aplicação node.js

```
var express = require('express');
```

```
var app = express();
app.get('/', function (req, res) {
res.send('Hello World!');
});
app.listen(3000, function () {
console.log('Example app listening on port 3000!');
});
```

Construa dentro de sua automação um processo de deploy e rollback seguro e rápido da aplicação node. O deploy e rollback deverá garantir a instalação das dependências node novas (caso sejam adicionadas ou alteradas a versão de algum dependência por exemplo), deverá salvar a versão antiga para possível rollback e reiniciar todos processos node sem afetar a disponibilidade global da aplicação na máquina.

  Resposta: Avaliei algumas alternativas, mas esta atividade ficou pendente.

A aplicação Node deverá ser acessado através de um Servidor Web configurado como Proxy Reverso e que deverá intermediar as conexões HTTP e HTTPS com os clientes e processos node. Um algoritmo de balanceamento deve ser configurado para distribuir entre os N processos node a carga recebida.

  Resposta: Foi utilizado o módulo HaProxy.

A fim de garantir a disponibilidade do serviço, deverá estar funcional uma monitoração do processo Node e Web para caso de falha, o mesmo deve reiniciar ou subir novamente os serviços em caso de anomalia.

  Resposta: Foi utilizado o módulo Pm2.

Desenvolva um pequeno script que rode um teste de carga e demonstre qual o Throughput máximo que seu servidor consegue atingir.

  Resposta: Foi utilizado o módulo Loadtest.

Desenvolva um script que parseie o log de acesso do servidor Web e deverá rodar diariamente e enviar por e-mail um simples relatório, com a frequência das requisições e o respectivo código de resposta (ex:5 /index.html 200).

  Resposta: O script foi elaborado utilizando o recurso de log do Proxy Reverso "HaProxy" e o programa "ssmtp" para simular o envio de e-mail em um ambiente real. Foi incluída uma tarefa no "Crontab" para execução desse script, todos os dias às 04h00.

Por fim; rode o seu parser de log para os logs gerados pelo teste de carga, garantindo que seu script terá perfomance mesmo em casos de logs com milhares de acessos.

  Resposta: Foi utilizado o recurso de log do Proxy Reverso "HaProxy".

---------- Processo de aplicação da configuração na máquina virtual "zerada" ----------

1. Instalar Sistema Operacional Linux Ubuntu 16.04 TLS Server

   No meu caso, instalei o Linux Ubuntu 16.04 TLS Server (Xenial), arquivo "ubuntu-16.04.4-server-amd64.iso", em um máquina virtual no Virtual Box instalado no meu Notebook e utilizei o Ubuntu Mate como interface gráfica (sudo add-apt-repository ppa:ubuntu-mate-dev/xenial-mate; sudo apt-install ubuntu-mate-core).

2. Criar usuário "anabasto" com acesso root via Sudo

3. Fazer login com usuário "anabasto"

4. Criar pasta git_repos em /home/anabasto

5. Instalar Nodejs (v8.11.2) e Npm (v5.6.0)

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
sudo apt-get install -y nodejs
```

sudo apt-get install npm

npm config set prefix 'nome da pasta npm-global'

Incluir caminho "nome da pasta npm-global/bin" no final da variável de ambiente $PATH, no respectivo script

6. Instalar Git

sudo apt-get install git-all

criar pasta /home/anabasto/git_repos/

7. Fazer clone do repositório https://github.com/anabasto/appnodejs1

8. Instalar a aplicação NodeJs

- Criar pasta /home/anabasto/myapp_nodejs1

- Copiar /home/anabasto/git_repos/appnodejs1/app_js1.js para /home/anabasto/myapp_nodejs1

- Entrar na pasta /home/anabasto/myapp_nodejs1:

    - Executar: npm init *** para gerar o arquivo package.json

    - Executar: npm install express *** para ativar e incluir a dependência do módulo Express no arquivo package.json

9. Instalar/Ativar HaProxy para a função de Proxy Reverso e balanceamento de carga para o site disponibilizado pela aplicação app_js1

npm -g install haproxy

*** Copiar arquivo "/home/anabasto/git_repos/appnodejs1/haproxy/haproxy.cfg"

10. Instalar Pm2 para a função de inicialização e monitoração da aplicação NodeJs

npm -g pm2

11. Passos para inicializar no boot a app_js1 pelo Pm2

cd <pasta-npm-global>/bin

sudo ./pm2 startup ubunt --service-name app_js1

sudo ./pm2 -i max start /home/anabasto/myapp_nodejs1/app_js1.js

sudo ./pm2 save

sudo systemctl restart app_js1

sudo ./pm2 list ** Na minha máquina que possui 2 processadores foi possível verificar as 2 instâncias da aplicação app_js1

12. Instalar SSMTP para simulação de envio de e-mail em ambiente real

- Criar pasta /home/anabasto/app_envio_emails

- Copiar arquivo "/home/anabasto/git_repos/appnodejs1/scripts/envia_relatorio_appjs" para a pasta /home/anabasto/app_envio_email

- Editar o arquivo de "crontab" do usuário "root" e incluir a linha abaixo:

    0 4 * * * /home/anabasto/app_envio_emails/envia_relatorio_appjs

13. Instalar Loadtest para teste de carga

npm install -g loadtest

14. Executar o teste de carga e acompanhar o log do HaProxy

    Exemplo: loadtest -c 2 --rps 20 http://127.0.0.1

    halog -H -st -q /var/log/haproxy.log