Macintosh HD:Users:alexandre:Desktop:LOGOS_DEPARTAMENTOS_PAPEL_DE_CARTA:CIENCIAS_ULISBOA_INFORMATICA_CMYK.eps



**Relatório de Situação do Projeto**(2ª Avaliação Periódica)

# Grupo 05

|  |  |  |
| --- | --- | --- |
|  | Nome | Número |
| 1 | André Silva | 58644 |
| 2 | Bernardo Caldas | 58650 |
| 3 | David Carreira | 58629 |
| 4 | Martim Emauz | 58668 |
| 5 | Miguel Martins | 58661 |
| 6 | Tomás Alves | 58633 |

# Implementação da arquitetura distribuída

A versão atualizada do [diagrama](../../../Downloads/PTI.drawio.png) vê refeitas grande parte das suas componentes. Da esquerda para a direita, a primeira mudança é a introdução da firewall que limita o tráfego entre os clientes e o sistema.

A sub-rede extnet é responsável pela comunicação VRRP entre os balanceadores de carga, bem como entre estes e os clientes externos. Através do *Google Compute Engine* (GCE), pudemos reservar um IP externo para que este seja estático e atribuído a apenas uma das máquinas de cada vez.

Na sub-rede default, comunicam os balanceadores de carga com os servidores web, e estes últimos comunicam com as instâncias que albergam a API. Através do algoritmo leastconn, garantimos que os pedidos são atribuídos ao servidor com menos conexões ativas, garantindo assim uma distribuição equitativa da carga.

A API comunica, com os servidores web, através de IPs internos na respetiva sub-rede, para maximizar a performance.

As bases de dados estão englobadas numa outra sub-rede database, que permite que comuniquem entre si, ou com a API.

O fluxo bidirecional da informação é transmitido entre todas estas componentes, atualmente, sem restrições de firewall, para evitar problemas técnicos que possamos encontrar.

Uma imagem com captura de ecrã, círculo

Descrição gerada automaticamente

Figura 1 - Diagrama de arquitetura

## Implementação dos balanceadores de carga

1. **Configuração de sub-redes**

Para começar, criámos uma sub-rede “***extnet***” para que os balanceadores de carga comuniquem com os clientes, de forma a isolar a comunicação externa e interna.

Esta foi configurada com um intervalo de endereços IPv4 **10.10.0.0/20**, e na mesma região em que são configuradas as máquinas virtuais. Adicionalmente, ativámos a opção que define que as *VMs* na sub-rede não têm IPs externos atribuídos, para que estes sejam atribuídos dinamicamente através do *Google Compute Engine* (GCE):

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 2 - Parâmetros de criação de um load balancer

1. **Criação de instâncias**

Criadas as redes, seguiu-se a criação das VMs responsáveis pelo balanceamento da carga. Criámos instâncias com nomes “load-balancer-X”, em que X corresponde ao número da ordem de criação dos balanceadores.

Olhando para o exemplo referido nos [slides](file:///C:\Users\Utilizador\Downloads\config_servers_PTI.pdf) disponibilizados relativamente aos Balanceadores, e através de pesquisa, utilizámos valores equivalentes aos sugeridos pelos docentes.

A **região escolhida** foi a mais barata, que nos permitisse ter proximidade ao público-alvo (Portugal) e máquinas disponíveis. Por isso, escolhemos criar as sub-redes e as *VMs* na Holanda.

Uma imagem com captura de ecrã, texto, file, preto

Descrição gerada automaticamente

Figura 3 - Região adotada

No que toca à configuração das máquinas, seguimos as diretrizes dadas nos exemplos dos docentes, no que toca à quantidade de processadores virtuais (*vCPU*) e de memória. Além disso, feita alguma pesquisa, verificámos que era adequada a configuração sugerida para o tipo de carga de trabalho que esperamos enfrentar com a execução do serviço [1]. Desta forma, os balanceadores suportarão tráfego TCP e HTTP com até **4000 conexões por segundo**.

Dito isto, estabelecemos a configuração da máquina do tipo E2-micro, que disponibiliza um máximo de **2 *vCPU***, com **1 *core***, e **1GB de memória**.

O **custo** torna-se **reduzido**, tanto pela escolha de região, como da configuração das máquinas:

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 4 - Configuração e preço estimado

O **sistema foi operativo** escolhido foi o ***Ubuntu***, com a versão 20.04 LTS, pois era a versão estável mais recente disponível equivalente à versão 18.04 referida pelos docentes. Trata-se de uma distribuição que suporta todo o software necessário para a implementação do nosso projeto, além de todos estarmos familiarizados com o seu funcionamento. O tamanho do disco é de **10GB**, um valor que além de recomendado pelos docentes, será mais do que suficiente para hospedar todo o código e software necessário para a sua implementação:

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

Figura 5 - Sistema operativo escolhido

Definimos que as máquinas virtuais terão acesso a **todas as *APIs*** da *Cloud*:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 6 - Permissões de acesso às APIs

Nas **definições avançadas**, e na aba de **Redes**, definimos o tipo de interfaces de rede como gVNIC, a interface de rede virtual da Compute Engine:

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Figura 7 - Configuração de interfaces de rede

As interfaces de rede implementam as sub-redes default, e extnet. A sub-rede default estabelecerá a comunicação entre os balanceadores de carga e os servidores web. A sub-rede extnet dará palco à comunicação entre os balanceadores e os pedidos provenientes dos clientes, bem como a comunicação VRRP entre ambos os balanceadores:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 8 - Configuração de interfaces de rede

Na interface de rede ***extnet***, configurámos o IP interno de ambas as máquinas para ser dinâmico. Este será o endereço que comunica com os servidores-web, direcionando o tráfego adequadamente entre as várias instâncias de servidores em vigor, através do algoritmo “***leastconn***”. O IP externo será um IP estático reservado, que será associado à máquina designada como mestre. Inicialmente, será associado ao balanceador de carga 1.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 9 - Configuração da sub-rede extnet

Na interface da rede ***default***, as máquinas terão IPs internos dinâmicos automaticamente designados, mas não terão IPs externos:

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 10 - Configuração da sub-rede default

## Implementação dos web servers

Criámos instâncias com o nome “servidor-X”, em que X corresponde ao índice do servidor sob a ordem da sua criação.

A **região escolhida** foi a mesma do que no caso dos balanceadores de carga, para garantir consistência:

Uma imagem com captura de ecrã, texto, file, preto

Descrição gerada automaticamente

Figura 11 - Região adotada

A **configuração das máquinas** no que toca à quantidade de vCPU e memória, foi idêntica à anteriormente referida:

O **sistema operativo** escolhido foi o mesmo do das máquinas dos balanceadores de carga:

Nas opções da **firewall**, ativámos as opções para permitir comunicação HTTP e HTTPS.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 12 - Definições de firewall

Nas **opções avançadas**, definimos, na aba de Redes, a configuração de performance de rede como *gVNIC* e a interface de rede como *default*:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 13 - Configuração de interface de rede

Na sub-rede default, as máquinas virtuais apenas terão a si associadas um IP externo cada uma.

Adicionalmente, verificámos a não-omissão da opção de *“auto-restart”* das máquinas, que é definida automaticamente.

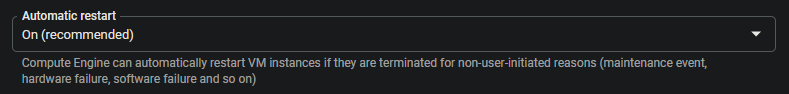


Figura 14 - Renício automático

## Implementação das bases de dados

1. **Configuração de sub-redes**

A sub-rede que permite que as bases de dados comuniquem entre si é a “***database***”. Esta foi criada sem quaisquer restrições na firewall, na gama de endereços IPv4 **10.11.0.0/20**. Esta foi também definida na mesma área das restantes componentes do sistema. As opções da configuração são semelhantes às da rede “*extnet*”

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 15 - Parâmetros da rede database

1. **Criação de instâncias**

À semelhança das outras máquinas virtuais, escolhemos um tipo de máquina e disco semelhantes. Ao contrário das outras, não ativámos a comunicação HTTP e HTTPS nestas instâncias.

Quanto às suas interfaces de rede, definimos IPs internos para que possam comunicar entre si. Um IP estático foi configurado para que as máquinas possam comunicar com serviços externos, como a API.

O Keepalived será configurado de forma que este endereço flutue entre as duas instâncias, caso uma destas falhe.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 16 - Configuração de redes

No sistema operativo destas, será instalado o **PostgreSQL**, que nos permitirá definir um IP externo para receber pedidos. Além disso, também nos dá a possibilidade de utilizar ***Streaming Replication***, que na prática copia os conteúdos de uma base de dados para a outra, à medida que estes são alterados.

# Escalabilidade

Elaborámos um script para automatizar e otimizar a inicialização dos servidores, que executa a parte das [instruções](file:///C:\Users\Utilizador\Downloads\instruções_web_servers.txt) necessárias para ter os servidores prontos. Em detalhe, este:

* Inicializa o modo *sudo*, para realizar as operações no sistema sem problemas com permissões de acesso, execução ou leitura;
* Atualiza a função “apt-get” do Linux, que nos deu problemas na instalação do apache nas primeiras tentativas, pois não encontra 4 recursos na instalação do mesmo;
* Instala o apache2, e emite a resposta “Y”, quando a instalação pergunta se deve proceder;
* Ativa e inicializa os serviços do apache2;
* Atualiza os conteúdos do ficheiro /var/www/html/index.html, Introduzindo “Servidor X” no título e num parágrafo na página HTML.Com recurso aos meta dados da própria instância, conseguimos recolher o nome dado à instância na GCP e, consequentemente, armazenar o número correspondente ao índice do servidor em questão. Com este número, podemos atualizar automaticamente os dados da página HTML referida, sem que ninguém tenha que o fazer manualmente.

O script, de momento, não funciona totalmente como pretendido.

Para instanciar os servidores, definimos alertas para determinados níveis de uso da memória e do processador. O software utilizado para este efeito é o **prometheus**. Através de pesquisa relativamente a sistemas com especificações semelhantes, definimos os seguintes valores para gerar uma nova instanciam conforme a percentagem da utilização de recursos durante um determinado tempo [2]:

* + - Utilização de **CPU**:
      * **75%** de uso por 10 ou mais minutos
      * **80%** de uso por 5 ou mais minutos
      * **85%** de uso por 1 ou mais minutos
      * **90%** de uso
    - Utilização de **memória**:
      * **75%** de uso por 2 ou mais minutos
      * **80%** de uso por 1 ou mais minutos
      * **85%** de uso

A [configuração](../../../Downloads/prometheusyml.txt) do prometheus define os [alvos](../../../Downloads/targets.txt), que neste caso serão os servidores web a monitorizar. O ficheiro “[*rules*](../../../Downloads/rules.txt)” define os *triggers* e implementa os valores referidos acima.

Adicionalmente, definimos *baselines* para os valores pretendidos de tempo de resposta. Através de uma pesquisa com o intuito de otimizar a performance e a saúde do sistema [3], determinámos o seguinte: Os pedidos HTTP devem demorar não mais do que **200ms** a serem respondidos; as chamadas à API devem demorar não mais do que **600ms**; As chamadas À aplicação devem demorar **1 segundo** ou menos.

Futuramente, utilizaremos o **ansible** para definir scripts funcionais que criem máquinas com os parâmetros necessários, e instalem o software necessário para a sua utilização. De momento, construímos apenas um [script](../../../Downloads/#!binbash.txt) provisório.

O objetivo é que, no final, possamos criar e apagar instâncias conforme o uso das mesmas.

# Tolerância a Faltas

Após a sua inicialização, ambas as máquinas responsáveis pelo balanceamento de carga foram configuradas para utilizar o ***HAProxy*** e o ***Keepalived***. Estes serão responsáveis pela propriedade de tolerância a falhas do sistema de balanceamento de carga, e pela distribuição do fluxo de processos **de forma justa** entre as várias instâncias de servidores web.

O ficheiro **/etc/haproxy/haproxy.cfg** define os parâmetros da configuração do *HAProxy*. Este será parecido nas duas instâncias, na sua estrutura e parâmetros:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 17 - Ficheiro de configuração do HAProxy

Após alguma pesquisa, definimos que o algoritmo de balanceamento da carga mais adequado para o nosso sistema seria o ***leastconn***. Este distribui os pedidos pelas instâncias com menos ligações existentes, garantindo que o fluxo é distribuído de forma **equitativa** [2] [3]. No que toca aos valores máximos para os períodos de inatividade das várias conexões (**timeout**), fizemos alguma pesquisa no que toca à sua otimização [4] [5], e consideramos os valores sugeridos pelos docentes ideais.

O ficheiro **/etc/keepalived/keepalived.conf** estabelece a relação entre as máquinas e o seu papel no caso de uma falha. A linha “***state***” define o estado da máquina mestre como “MASTER” e o estado da máquina de reserva como “*BACKUP*”. O parâmetro “***priority***” estabelece a importância relativa das instâncias. A instância mestre terá um valor de 100, enquanto a outra terá um valor igual a 50.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 18 - Ficheiro de configuração do keepalived

O ***script*** **etc/keepalived/takeover.sh** define as ações a cumprir se uma máquina deixar de verificar que a outra está funcional. Este envia comandos à *Google Cloud* para alterar a configuração da máquina alheia, removendo a associação desta ao IP externo, e associa-o a si mesma.

As bases de dados também utilizarão um modo de disponibilidade parecido, para evitar falhas.

# Bibliografia

[1] HAProxy Enterprise, 'Install HAProxy Enterprise on Linux' - Disponível em: https://www.haproxy.com/documentation/haproxy-enterprise/getting-started/installation/linux/

[2] Deltablue, 'Optimizing HAProxy for security and performance by tuning timeouts ' - Disponível em: https://delta.blue/blog/haproxy-timeouts/.

[3] Solarwinds, 'HAProxy Logging- How to Tune Timeouts for Performance' - Disponível em: https://www.papertrail.com/solution/tips/haproxy-logging-how-to-tune-timeouts-for-performance/.

[4] Digital Ocean, 'An Introduction to HAProxy and Load Balancing Concepts' - Disponível em: https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts#leastconn

[5] Serverfault, 'HA Proxy - roundrobin vs leastconn' - Disponível em: https://serverfault.com/questions/457506/ha-proxy-roundrobin-vs-leastconn