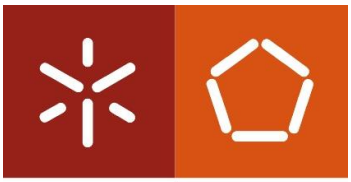

Universidade do Minho

Departamento de Informática

Aplicações e Serviços de Computação em Nuvem

Trabalho Prático

Grupo Nº 31

Maria Marques (PG47489)
Pedro Pereira (PG47584)

16 de fevereiro de 2022

Conteúdo

Capítulo 1

Introdução

A tecnologia tem cada vez mais um papel preponderante no nosso mundo, pelo que esta tem de ser capaz de se adaptar e responder com sucesso às necessidades de qualquer tipo de cliente. Para que isto aconteça, são necessárias infraestruturas seguras, confiáveis e eficientes, que estejam preparadas para responder aos desafios no âmbito de tolerância a falhas, escalabilidade, alocação de recursos, disponibilidade, entre outros.

Este trabalho prático tem como objetivo principal a caracterização, análise, instalação, monitorização e avaliação da aplicação **Wiki.js** (https://js.wiki). Este pretende ainda aplicar os conceitos lecionados na unidade curricular de forma a garantir a automatização completa do processo de instalação, monitorização e avaliação da aplicação.

Capítulo 2

Arquitetura e Componentes da Aplicação

A aplicação **Wiki.js** utiliza uma arquitetura simples de cliente-servidor, sendo composta por um cliente Web (*frontend*) e ainda um servidor **Wiki.js** e uma base de dados (que juntos constituem a *backend*) de forma a suportar o funcionamento desta.

Nesta arquitetura inicial da aplicação, é identificado primariamente um problema de disponibilidade, uma vez que não existe qualquer tipo de replicação, sendo que no caso de algum dos 2 componentes da *backend* falhar, o serviço **Wiki.js** fica automaticamente indisponível. Isto torna o sistema vulnerável a qualquer eventualidade, o que é extremamente indesejável quando se pretende ter a maior disponibilidade possível.

Para além disso, é identificado um ponto crítico de performance na ligação entre o servidor **Wiki.js** e a sua base de dados, uma vez que é nesta última que ficarão armazenados todos os dados persistentes do sistema, e consequentemente será utilizada sempre que o serviço for acedido por um cliente, por exemplo para obter o conteúdo de cada página ou para efetuar login.

Capítulo 3

Instalação Automática da Aplicação

De forma a efetuar a instalação automática e configurável da aplicação decidimos utilizar a ferramenta Ansible, uma vez que era esta com que o grupo se encontrava mais familiarizado, tendo sido utilizada nas aulas teórico-práticas desta unidade curricular, e por se tratar de uma ferramenta com um leque extenso de funcionalidades, tendo inclusivamente integrada um módulo especificamente para a utilização da *Google Cloud Platform*. Esta permitiu-nos então a automatização da criação de todos os componentes da *GCP* a ser utilizados, bem como da sua configuração.

Para esta instalação automática deverá ser utilizada a tag *create*, sendo que também é possível parar a execução das instâncias dos servidores **Wiki.js** utilizando a tag *stop*. Optamos pela não utilização de containers por uma questão de simplicidade, e pelo facto da arquitetura da **Wiki.js** não ter uma granularidade que torne benéfica a utilização de containers em relação a máquinas virtuais.

3.1 Criação de Servidor Wiki.js e Base de Dados

Para a criação dos servidores aplicacionais e de base de dados foi utilizado extensivamente o módulo da *GCP*, uma vez que foi necessária a criação de uma firewall para permitir o acesso externo aos servidores, bem como discos e endereços para as máquinas virtuais, e obviamente as máquinas virtuais de cada componente em si.

As instâncias do servidor **Wiki.js** são criadas de modo configurável, sendo possível num dos ficheiros de configuração definir o número de instâncias a ser criadas. Estas são colocadas numa *Target Pool* que em conjunto com uma *Forwarding Rule* resulta no balanceamento da carga pelas instâncias criadas dos pedidos recebidos no endereço definido na *Forwarding Rule*. Estas instâncias de máquinas virtuais têm as seguintes especificações: 2 vCPUs (Intel Broadwell), 8GB de memória RAM, 10GB de armazenamento e corre Ubuntu 20.04.

Foi utilizado o serviço de SQL da *GCP*, que permitiu a criação de uma instância de uma base de dados **PostgresSQL**, sendo configurada automaticamente aquando da sua criação. Este inclui um serviço de alta disponibilidade automático, que utiliza uma instância primária e uma instância standby, cada

uma com o seu disco persistente, sendo então feita replicação. Esta instância tem as seguintes especificações: 2 vCPUs, 7.5 GB de memória RAM e 10 GB de armazenamento (com aumento automático)

Capítulo 4

Monitorização

Para a monitorização foi utilizado a stack **ELK**. Esta utiliza Beats para a observação dos eventos no sistema, Logstash para a coleção dos dados, Elasticsearch para a análise dos dados obtidos e Kibana para a apresentação. Estas ferramentas de monitorização podem ser instaladas e configuradas automaticamente na aplicação através da utilização da tag 'monitoring' na execução do playbook Ansible elaborado, sendo criada uma máquina virtual para atuar como servidor monitor, correndo esta o Elasticsearch e Kibana. Cada instância de servidor da **Wiki.js** irá correr Metricbeats a reportar esses dados para o servidor monitor.

As principais métricas que podem ser visualizadas são a utilização de CPU, a carga no sistema (média móvel), a utilização de memória e o tráfego de rede em número de pacotes e em bytes, para cada uma das instâncias. É ainda possível visualizar algumas métricas agregadas das instâncias, como utilização de CPU, memória, disco e tráfego de rede. Consideramos que estas métricas sejam suficientes para uma monitorização dos servidores **Wiki.js**, uma vez que permitem a observação em tempo real da carga que os servidores.

Capítulo 5

Testes

De modo a avaliar o desempenho da nossa instalação recorremos à realização de alguns testes utilizando o JMeter. Os testes permitiram avaliar a aplicação em causa de forma realista através da simulação de diferentes comportamentos de utilizadores. Os testes foram corridos na linha de comandos de modo a que o valores obtidos nos testes sejam o mais corretos possível, evitando o overhead da GUI.

5.1 Teste 1

O teste visa representar um cenário em que um utilizador acede à página inicial do **wiki.js**. Este é representativo de grande parte da utilização deste serviço, uma vez que este consiste primariamente na visualização de páginas.

Ao correr este teste será testado o seguinte método:

1. GET /

O teste foi corrido com 200, 1000, 2000 e 5000 threads, sendo que recorremos aos relatórios gerados pelo JMeter para apresentar os seguintes resultados.

Threads	Erro	Mediana do Tempo Resposta (ms)	Throughput (Trans. /s)
200	0.00%	136	53.4
1000	0.00%	237	142.7
2000	0.00%	266	341.3
5000	0.00%	2171	651.3

Figura 5.1: Teste 1

Tal como podemos observar nos dados obtidos, em todos os testes elaborados, obtivemos uma percentagem de erro de 0.00%. Quando olhamos para a mediana dos tempos de resposta, verificamos que os servidores **Wiki.js** vão progressivamente ficando sobrecarregados, mas mantendo sempre níveis razoáveis de tempos de resposta. Relativamente à taxas de transferência, podemos concluir que tal como o tempo de resposta, esta é maior consoante o aumento do número de threads.

5.2 Teste 2

Este teste visa representar um cenário em que um utilizador acede à página principal, efetua login e retorna à página inicial. Este teste consiste nos seguntes passos:

1. GET /
2. GET /login
3. POST /graphql
4. GET /

O teste foi corrido com 200, 1000 e 2000 threads, sendo que recorremos aos relatórios gerados pelo JMeter para apresentar os seguintes resultados.

Threads	Erro	Mediana do Tempo Resposta (ms)	Throughput (Trans. /s)
200	0.00%	1665	1.3
1000	0.00%	2913	154.2
2000	0.00%	3475	161.9

Figura 5.2: Teste 2

Tal como observado anteriormente, com o aumento substancial de pedidos, o atraso na resposta aumenta também de forma considerável, mas mantendo sempre níveis razoáveis de resposta.

No entanto, as percentagens de erro devem ser descartadas uma vez que o **Wiki.js** quando sobrecarregado, envia uma resposta HTTP 200 OK, mas com mensagem de erro "Too many requests, please try again in X seconds", que o JMeter não interpreta como erro devido ao tipo da resposta HTTP.

Capítulo 6

Conclusões

A elaboração deste projeto permitiu aos elementos do grupo de apurar e evoluir as suas competências, através da consolidação dos conceitos aprendidos na unidade curricular de Aplicações e Serviços de Computação em Nuvem, mais concretamente, os métodos e ferramentas que devem ser aplicados à resolução do problema de planear, operar e avaliar uma infraestrutura de elevada disponibilidade e desempenho.

Apesar das dificuldades sentidas em alguns pontos do desenvolvimento, foi possível concluir o trabalho previsto com sucesso, desenvolvendo uma estrutura que além de disponível e com razoável performance relativa, apresenta também um nível aceitável de escalabilidade.

Para concluir, o grupo considera que foi elaborado um trabalho razoável, que consegue a instalação de todos os componentes necessários para a utilização da **Wiki.js** bem como a sua monitorização de forma automática com apenas um único comando. No entanto, estão claramente identificados pontos onde ainda se poderiam apresentar melhorias tais como a execução de diferentes componentes da aplicação em containers , uma vez que estes podem ser benéficos pela redução do seu overhead em relação às máquinas virtuais. Podiam ainda ter sido utilizadas de outras ferramentas de coleção de dados como Packetbeat para medir tráfego de rede ou Heartbeat para monitorizar a disponibilidade das instâncias, de forma a conseguir seguir mais pormenorizadamente o estado destas e o seu desempenho. Para além de todos estes pontos, podia ainda ter sido efetuada a automatização dos testes realizados, bem como a exploração dos testes realizados de forma a melhorar a sua avaliação e ainda o desenvolvimento de novos testes mais complexos.