



Universidade do Minho
Escola de Engenharia

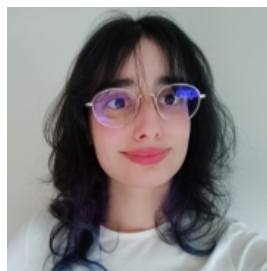
Aplicações e Serviços de Computação em Nuvem

Grupo 23

2023/2024

Tomás Cardoso Francisco pg54263
Afonso Xavier Cardoso Marques pg53601
Maria Eugénia Bessa Cunha pg54042
Ana Filipa da Cunha Rebelo pg53624

Dezembro 2023



Índice

1	Introdução	4
2	Laravel.io	4
3	Arquitetura	5
4	Automatização	6
4.1	Criação da imagem <i>Docker</i>	6
4.2	Criação do <i>Cluster Kubernetes</i>	6
4.3	Criação das VMs	6
4.4	Instalação do <i>Laravel.io</i>	7
5	Métricas de avaliação	8
6	Avaliação e Testes	11
6.1	Considerando a instalação base proposta pelo grupo para a Tarefa 1:	11
6.2	Com base nas respostas dadas para as questões anteriores:	11
7	Conclusão	13

List of Figures

1	Logótipo da Laravel	4
2	Arquitetura proposta	5
3	Cluster	6
4	Nodos	7
5	Carga de Trabalho (<i>Pods</i>)	7
6	Serviços	7

1 Introdução

Neste trabalho realizado no âmbito da UC de Aplicações e Serviços de Computação em Nuvem, foi proposta a automatização da instalação da aplicação **Laravel.io**. Complementarmente, também foi efetuada a sua caracterização, análise, monitorização e avaliação.

Deste modo, o presente relatório pretende expôr todo o trabalho efetuado, detalhando todos os passos que compuseram a sua realização e justificações das decisões tomadas pelo grupo.

A primeira etapa do trabalho consistiu no *deployment* da aplicação com recurso ao *Ansible* e aos serviços fornecidos pelo *Google Cloud Platform*, efetuando a instalação no menor número de passos possível. A segunda fase do trabalho consistiu no *deployment* da aplicação recorrendo aos *kubernetes* e aos serviços suportados pelo *Google Cloud Platform* com o intuito de otimizar o desempenho, escalabilidade e resiliência da aplicação. Por fim, a terceira e última etapa do trabalho envolve a seleção, instalação e testes automatizados das ferramentas de monitoramento e avaliação. Isso inclui escolher as melhores ferramentas, instalá-las e realizar testes automáticos que simulem a interação dos utilizadores.

2 Laravel.io

O **Laravel.io** é uma comunidade *online* essencial para desenvolvedores que trabalham com o *framework PHP*, *Laravel*. Funciona como um fórum ativo, onde os utilizadores compartilham conhecimento, encontram soluções para problemas, acedem a recursos educativos e se mantêm atualizados sobre eventos e notícias relevantes para o *Laravel*. Além disso, oferece oportunidades de colaboração em projetos, *networking* e é um espaço aberto para aprendizagem contínua. É uma plataforma vital para a comunidade de desenvolvedores *Laravel*.



Figure 1: Logótipo da Laravel

3 Arquitetura

A arquitetura do sistema adota um paradigma multicamada, caracterizado pela estruturação em diferentes níveis funcionais. Essa abordagem compreende diferentes camadas, das quais se destacam a interface destinada ao utilizador e uma camada dedicada à persistência dos dados. Tal configuração proporciona uma clara separação de responsabilidades e funcionalidades entre os elementos do sistema, permitindo a interação eficiente entre a interface de utilizador e os processos de armazenamento e manipulação de dados. Esta estrutura *multilayer* não apenas facilita a manutenção e evolução do sistema, mas também promove uma maior flexibilidade e escalabilidade, garantindo uma experiência robusta e adaptável aos requisitos e demandas do ambiente operacional.

Dito isto, para atender aos requisitos estabelecidos, além dos componentes já mencionados, é necessário introduzir os seguintes elementos:

- *Técnicas Google*

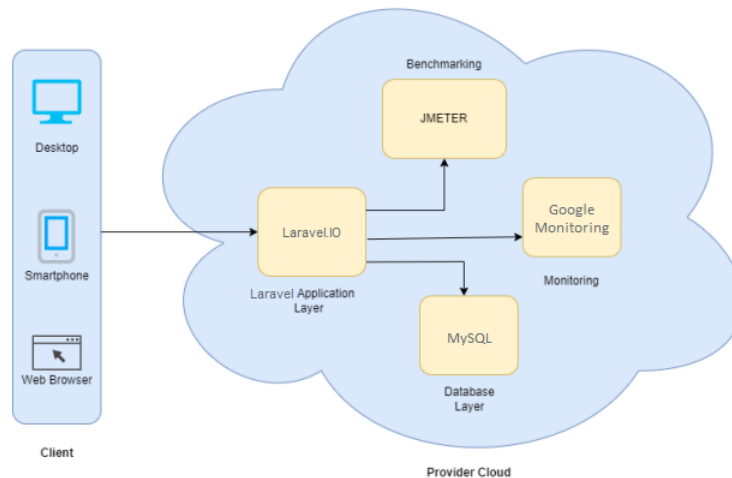


Figure 2: Arquitetura proposta

- JMeter - é uma ferramenta de auxílio na realização de testes de carga a sistemas computacionais.
- Google Monitoring-é uma plataforma de monitoramento oferecida pelo Google Cloud Platform (GCP). Essa ferramenta oferece uma visão abrangente do desempenho e da saúde de várias camadas de uma aplicação hospedada no GCP.

e pra alem disso falta expandir -m

4 Automatização

A automatização do processo de deployment da aplicação **Laravel.io** utilizou as ferramentas *GKE* (*Google Kubernetes Engine*) e *GCP* (*Google Cloud Platform*), através de *playbooks* e da linguagem *Ansible*. Nas seguintes secções encontra-se a explicação de todo o processo de instalação e *deployment* das várias componentes utilizadas no projeto.

4.1 Criação da imagem *Docker*

A primeira etapa do trabalho consistiu na criação da imagem docker para a aplicação Laravel.io. Para tal o grupo criou um Dockerfile que descreve passo a passo as configurações e instalações necessárias para criar um ambiente funcional para a aplicação começando com a definição da imagem base como Ubuntu 22.04 e procedendo com as seguintes etapas:

- Atualização e Instalação de Dependências: As instruções RUN foram utilizadas para atualizar o sistema, instalar o PHP 8.2, Node.js, Composer, Git, MySQL Client e outras dependências essenciais.
- Configuração do Ambiente Laravel: O Dockerfile clona o repositório da aplicação Laravel.io, configura o arquivo de ambiente .env e gera uma chave única para a aplicação Laravel usando o comando `php artisan key:generate`.
- Gestão de Dependências da Aplicação: As dependências do Laravel foram geridas usando o Composer e o Node Package Manager (npm). O Dockerfile executa os comandos `composer install` para as dependências PHP e `npm install` `npm run build` para as dependências JavaScript da aplicação.
- Configuração Final e Exposição de Porta: Um script de inicialização foi copiado para dentro da imagem e configurado como o comando padrão a ser executado quando a imagem é iniciada. Além disso, a porta 80 foi exposta para permitir o acesso à aplicação.

4.2 Criação do *Cluster Kubernetes*

A fase inicial do trabalho envolveu a criação de um *cluster Kubernetes* com um número arbitrário de nodos. Ao criar uma instância de *cluster* no *Google Cloud*, ocorre automaticamente a geração de máquinas virtuais que sustentam os nodos vinculados ao referido *cluster*.

rever tudo isto -m

Para realizar essa etapa, foi utilizado um ficheiro *playbook* que emprega o módulo *GCP Container Cluster* para efetuar a criação do *cluster*, e o módulo *GCP Container Node Pool* para realizar a criação dos nodos associados a esse *cluster*.

<input type="checkbox"/>	Status	Name ↑	Location	Fleet ?	Number of nodes	Total vCPUs	Total memory
<input checked="" type="checkbox"/>	✓	ascn-cluster	us-central1-a	REGISTER	2	4	4 GB

Figure 3: Cluster

4.3 Criação das VMs

Se desejarmos criar o *cluster Kubernetes* a partir de uma máquina virtual (VM) em vez do *localhost*, temos a possibilidade de automatizar a criação de uma instância de máquina virtual na *Google Cloud*. Para alcançar

esse objetivo, desenvolvemos um *playbook* que realiza a criação de um número arbitrário de VMs, seguindo as configurações indicadas, tais como tipo de máquina, imagem, zona, projeto e tamanho do disco.

rever isto outra vez com o código -m

É importante salientar que a criação das VMs exigiu a utilização do módulo *'gcp compute instance'* do *Ansible*. Além disso, para executar com sucesso a criação das VMs na plataforma *Google Cloud*, foi necessário levar em consideração a chave do projeto da *ASCN*, obtida na *Google Console*, a fim de fornecer acesso ao projeto.

Todo o processo de criação das VMs foi realizado de forma remota, utilizando um *host* externo à *Google Cloud*. Esse método permitiu a execução eficiente das tarefas planejadas, facilitando a gestão e configuração das instâncias virtuais de maneira flexível e adaptada às necessidades específicas do projeto.

Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	gke-ascn-cluster-default-pool-3e7d1994-02r9	us-central1-a		gke-ascn-cluster-default-pool-	10.128.15.216 (nic0)	34.135.91.140 (nic0)	SSH ▾
✓	gke-ascn-cluster-default-pool-3e7d1994-p0zv	us-central1-a		gke-ascn-cluster-default-pool-	10.128.15.215 (nic0)	34.41.175.42 (nic0)	SSH ▾

Figure 4: Nós

4.4 Instalação do *Laravel.io*



Name ↑	Status	Type	Pods	Fleet ?	Namespace	Cluster
laravel-deployment	✓ OK	Deployment	1/1	 ascn-23-24	laravel	ascn-cluster
mysql-deployment	✓ OK	Deployment	1/1	 ascn-23-24	db	ascn-cluster

Figure 5: Carga de Trabalho (*Pods*)


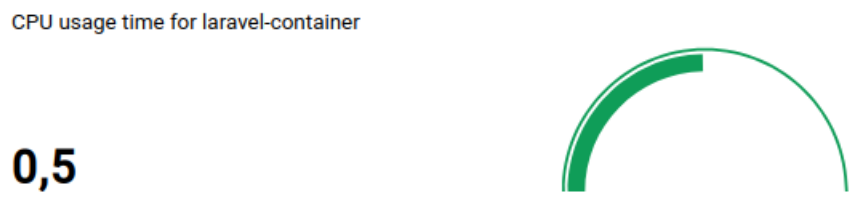
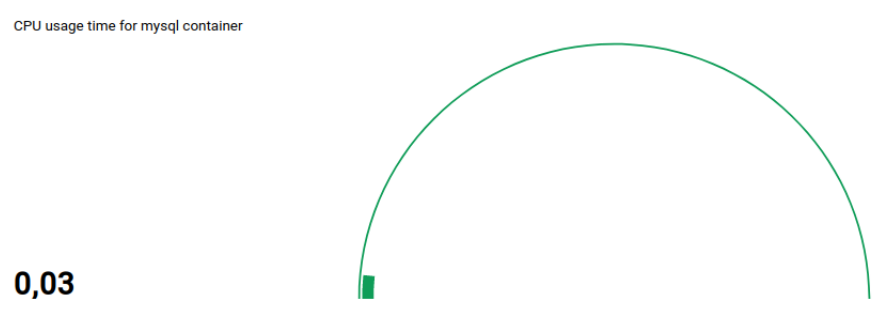
Name ↑	Status	Type	Endpoints	Pods	Namespace	Clusters
laravel-service	✓ OK	External load balancer	34.173.218.185:80 	1/1	laravel	ascn-cluster
mysql-service	✓ OK	Cluster IP	10.0.9.37	1/1	db	ascn-cluster

Figure 6: Serviços

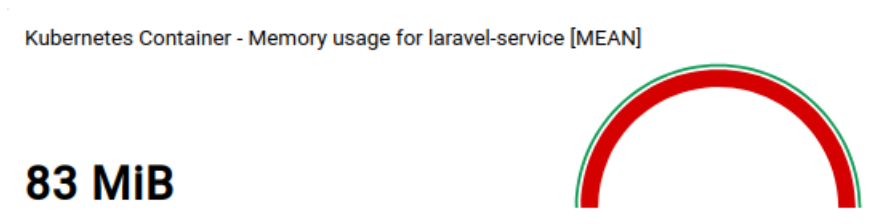
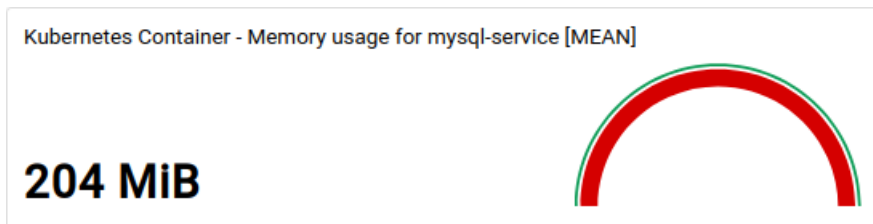
5 Métricas de avaliação

Tendo em conta o problema e as métricas disponibilizadas pelo Google Monitoring, resolvemos utilizar as seguintes métricas, sendo que as que se considerou mais pertinentes no problema atual:

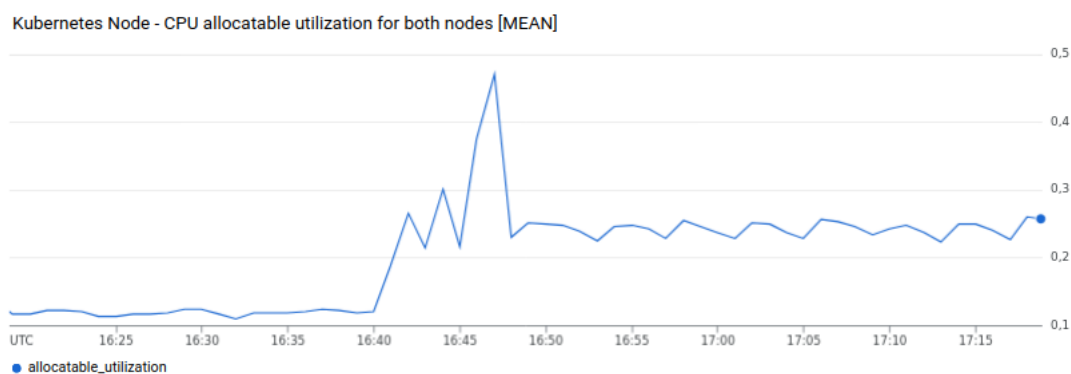
- Tempo de utilização do CPU nos containers do MySQL e Laravel



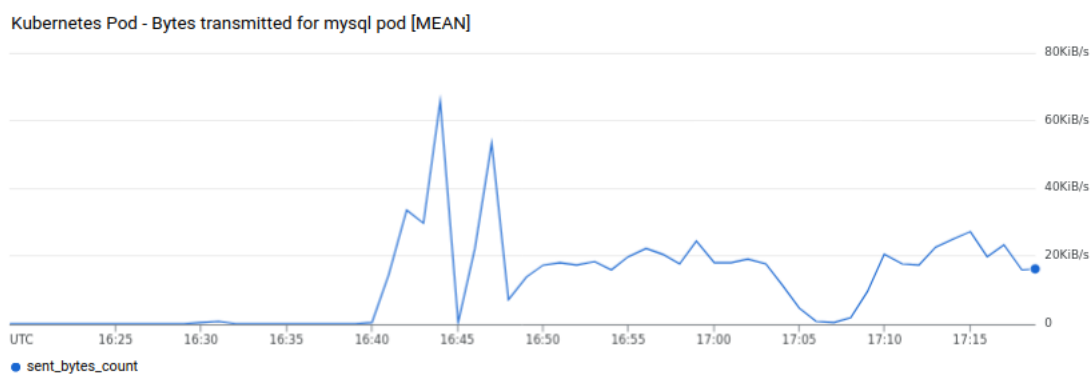
- Utilização de memória pelos containers MySQL e Laravel



- Utilização do CPU para os nodos do cluster



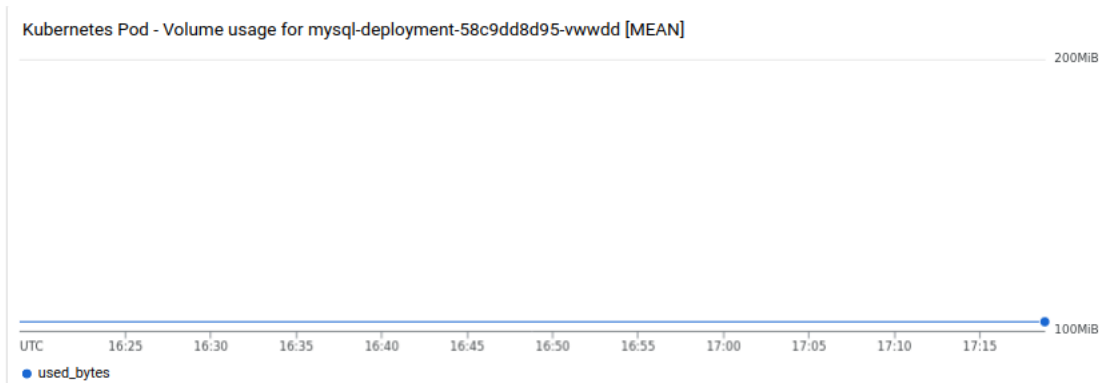
- Bytes transmitidos do MySQL Pod



- Bytes recebidos do MySQL Pod



- Volume usado do MySQL Pod



6 Avaliação e Testes

Com vista a poder responder às questões do enunciado e para poder efetuar testes de desempenho da implementação desenvolvida, foi necessário criar testes que permitissem avaliar os diferentes componentes e funcionalidade da aplicação, tendo em conta a possível variação de usuários que possam estar a utilizar. Para tal, recorreu-se à ferramenta JMeter, que permite efetuar testes de carga/stress através de um ficheiro de input, assim como simular um grande fluxo de utilizadores na aplicação.

Posteriormente, já com os ficheiros de teste obtidos, abriu-se os ficheiros no JMeter e iniciou-se os testes, sendo atribuídos diferentes valores de threads na aba "Thread Group", permitindo assim submeter a plataforma a uma maior ou menor carga. É de reforçar avaliação experimental da aplicação foi combinada com as métricas de monitorização, permitindo assim extrair observações mais completas e incisivas sobre os resultados observados, como será demonstrado em seguida.

6.1 Considerando a instalação base proposta pelo grupo para a Tarefa 1:

1. Para um número crescente de clientes, que componentes da aplicação poderão constituir um gargalo de desempenho?

R: Para responder a esta questão implementamos um playbook de ansible que permite correr um teste onde são lançadas multiplas threads que acedem ao serviço do Laravel. Para o primeiro teste, começou-se por 100 threads e obteve-se valores ligeiramente superiores aos valores de referência, demonstrando uma elevada capacidade para este valor.

2. Qual o desempenho da aplicação perante diferentes números de clientes e cargas de trabalho?

R:

3. Que componentes da aplicação poderão constituir um ponto único de falha?

R:

6.2 Com base nas respostas dadas para as questões anteriores:

1. Que otimizações de distribuição/replicação de carga podem ser aplicadas à instalação base?

R:

2. Qual o impacto das otimizações propostas no desempenho e/ou resiliência da aplicação?

R:

7 Conclusão

Após a conclusão do trabalho prático, apresentamos uma análise crítica, ponderada e refletida sobre o projeto.

rever isto please -m

É relevante salientar que a implementação do Laravel está funcional e operacional, destacando-se o *deployment* eficiente através das funcionalidades providenciadas pelo *Google Cloud*, nomeadamente o *Google Kubernetes Engine (GKE)*. Sublinhamos também a utilização de recursos como o balanceamento de carga através de um *LoadBalancer*, que contribui significativamente para a estabilidade e desempenho do sistema.

No que concerne a melhorias e atualizações no projeto, consideramos benéfico implementar uma hierarquia na base de dados. Propomos a criação de uma base de dados primária responsável por suportar o programa, com bases de dados secundárias a entrar em funcionamento em caso de falha da base primária. Esta abordagem proporcionaria maior robustez e resiliência ao sistema, assegurando uma resposta contínua face a eventuais falhas.

Resumindo, acreditamos que o trabalho tem um balanço positivo. As dificuldades encontradas foram superadas com sucesso, e todos os requisitos propostos foram integralmente cumpridos.