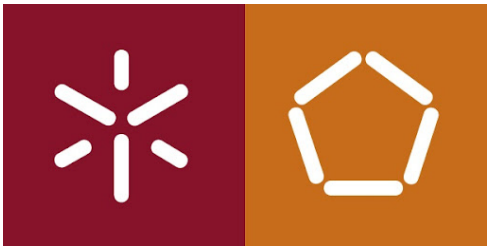

Universidade do Minho

Mestrado em Engenharia Informática

Deployment da aplicação *Wiki.js*

Aplicações e Serviços de Computação em Nuvem

Grupo 50

Gustavo Lourenço (pg47229)
Leonardo Marreiros (pg47398)
Martim Almeida (pg47514)
Pedro Fernandes (pg47559)

11 de janeiro de 2022

Conteúdo

Lista de Figuras

1 Introdução

O Trabalho prático consiste no *deployment* da aplicação *Wiki.js* através dos conceitos adquiridos na UC, com o objetivo que este seja realizado de forma mais automatizada possível.

Para isso, ao longo do presente relatório serão abordados os procedimentos quer para a automatização da instalação da aplicação, quer para a sua monitorização e avaliação. O projeto em si, encontra-se dividido nas seguintes 4 tarefas:

- **Tarefa 1:** Compreender a arquitetura, os componentes da aplicação e o seu funcionamento;
- **Tarefa 2:** Realizar a instalação automática (e que também possa ser posteriormente configurável) da aplicação na *Google Cloud Platform* através de uma ferramenta de *Provisioning*;
- **Tarefa 3:** Escolher uma ferramenta de monitorização e definir quais métricas a monitorizar da aplicação;
- **Tarefa 4:** Escolher uma ferramenta de avaliação, na qual seja possível executar os testes de forma automática e configurável.

2 Tarefa 1

Nesta tarefa, de forma a compreender quais são os componentes fulcrais da aplicação *Wiki.js*, foi realizada uma pesquisa à cerca do tema na documentação da aplicação em questão.

Os requisitos apresentados são:

- É recomendado que o *CPU* da máquina na qual o *Wiki.js* será executado tenha pelo menos 2 *cores*, pelo menos 1GB *RAM* disponível e ainda que possua pelo menos 1GB de Armazenamento;
- A aplicação recorre a uma base de dados, na qual a recomendada é *PostgreSQL*, sendo esta a que apresenta melhor perfomance e compatibilidade tem com a aplicação.
- Para ser executada aplicação necessita necessariamente de *Node.js*. No caso de numa tarefa seguinte se opte por usar o *container* disponível do *Wiki.js*, esta mesma já terá o *Node.js* incluído.

Com isto, podemos verificar que esta aplicação tem um padrão de distribuição do tipo *Client-Server*, uma vez que todos os dados e funcionalidades da aplicação se encontram presentes na máquina onde será realizado o *deployment*.

3 Tarefa 2

Esta tarefa consta no desenvolvimento dos vários *Scripts* e *Recipes* necessárias para realizar o *deployment* automático na *Google Cloud Platform*.

Como abordado, nas aulas práticas, a ferramenta de *provisioning* utilizada nesta trabalho foi *Ansible* devido à sua vasta coleção de Módulos presentes na ferramenta.

Esta tarefa tem como requisito a possibilidade dos diferentes componentes da aplicação correrem em *containers* diferentes.

Visto que o *Wiki.js* fornece uma *Docker Image* da aplicação e a biblioteca de imagens do *Docker* contém imagens das várias versões do *PostgreSQL*, optamos por neste trabalho usar containers.

A utilização de *containers* na nossa abordagem desta tarefa foi motivada pelas vantagens que os *containers* fornecem quanto à sua portabilidade, ao seu isolamento e ao empacotamento das aplicações e das suas dependências, tal como apresentado na tarefa anterior sobre o *container* do *Wiki.js*.

Também é pedido que seja possível aumentar o número de réplicas de um componente de forma automática e configurável. Uma vez que a ferramenta *Kubernetes*, apresentada nas aulas práticas, dispõe dessa disponibilidade, a nossa aplicação irá executar num *Cluster* do Kubernetes na *Google Cloud Platform*, tirando partido do *Google Kubernetes Engine*. Neste *cluster*, é possível realizar a replicação da base de dados, sendo este o componente principal para ser replicado, uma vez que contém toda a informação apresentada na aplicação, desde os dados dos utilizadores até às páginas geradas pelos mesmos dentro da aplicação.

3.1 *Deployment da aplicação*

Para realizar o *deployment* da aplicação, tentando minimizar o número de passos manuais possíveis, este processo foi dividido em três partes:

1. Criação de uma Máquina Virtual na *Google Cloud Plataform* através de uma máquina local;
2. Através da Máquina Virtual criada no passo anterior, realizar o *deployment* da rede, do *cluster* e do *node pool* do *Kubernetes*;
3. *Deployment* da aplicação para o Kubernetes;

3.1.1 Criação da Máquina Virtual

Para gerar a máquina virtual, utilizamos uma *Recipe* do *Ansible*. Posteriormente, através do comando *scp* enviamos as restantes *recipes* necessárias para o *deployment* do *Ku-*

bernetes. Em seguida é corrido um *script* que irá instalar na máquina virtual todas as dependências que esta necessita para realizar o *deployment*, isto é, adicionar o repositório do *ansible*, em seguida atualizá-lo, instalá-lo e por fim autenticar-se na *Google Cloud Plataform*. Por fim, é corrido um *playbook* que trata da instalação do *Kubernetes* assim como das dependências de que este necessita para funcionar corretamente.

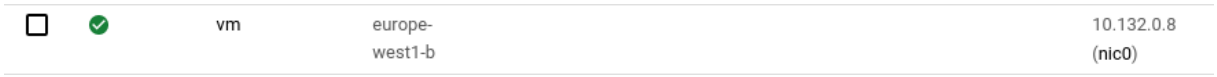

Figura 1: Máquina Virtual gerada

3.1.2 Criação do Cluster, da Rede e da Node Pool

Este processo é realizado através de uma *recipe* na qual é gerado um *cluster*, com uma rede virtual de comunicação entre os vários *Nodes*. O *cluster* inicialmente contém 3 nodos, sendo possível aumentar este número modificando uma variável presente na *recipe* mencionada anteriormente.

Em adição, foi também criada uma *recipe* que torna possível a destruição deste cluster, tornando também esta tarefa automática.

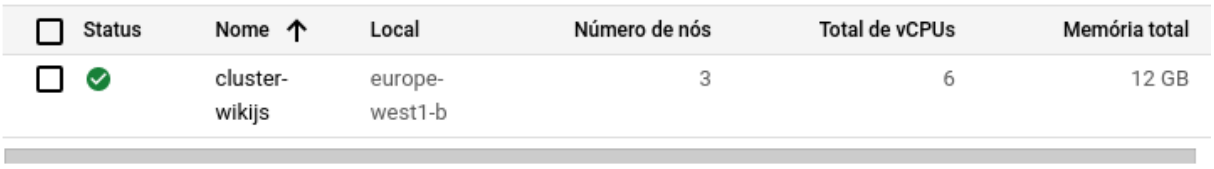

Figura 2: *Cluster*

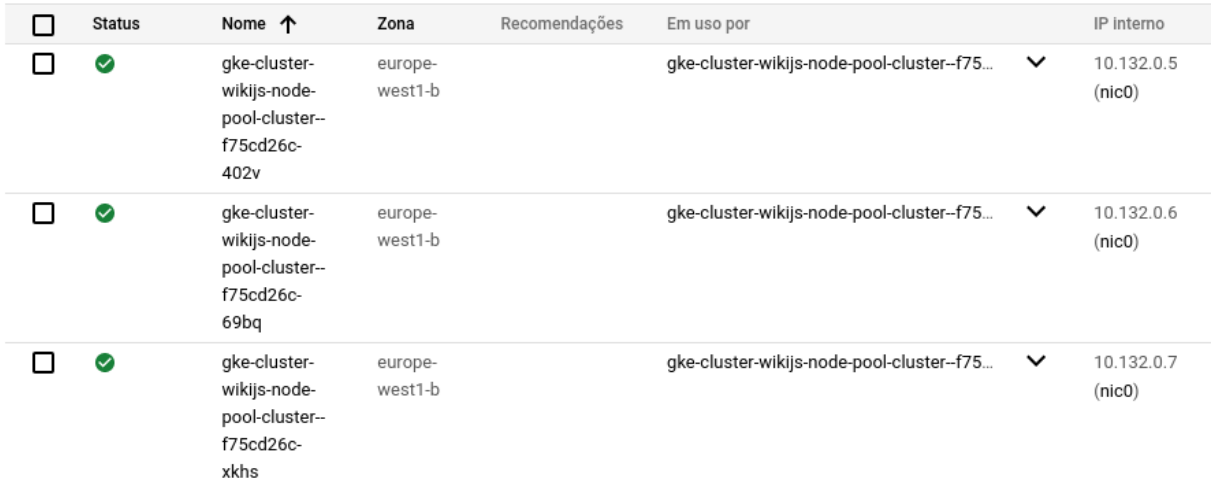

Figura 3: *Nodes* do *Cluster*

3.1.3 *Deployment* da Aplicação

Tal como na parte anterior, também este processo é realizado através de uma *recipe*.

Esta *recipe* inicialmente gera o seguinte:

- **Um *namespace*** próprio para a aplicação;
- **Um *StorageClass*** na qual ficará armazenada a base de dados e onde a *Pod* do *PostgreSQL* dará *claim*;
- **Dois serviços**, um para a base de dados e um outro para o *Wiki.js*. O serviço da aplicação *Wiki.js* é do tipo *LoadBalancer*, tornando possível o acesso à mesma de modo remoto.
- **Os Pods** da base de dados e da aplicação tornando possível a pronta utilização da aplicação.

Nome ↑	Fase	Volume	Classe de armazenamento	Namespace	Cluster
app-storage-postgres-0	Bound	pvc-778349bf-6c5e-461a-852c-7f909b9cede9	gold	wikijs	cluster-wikijs

Figura 4: *StorageClass* do *Cluster*

Nome ↑	Status	Tipo	Pontos de extremidade	Pods	Namespace	Clusters
postgres	OK	IP do cluster	Nenhum	1/1	wikijs	cluster-wikijs
wikijs	OK	Balanceador de carga externo	35.240.12.133:3000	1/1	wikijs	cluster-wikijs

Figura 5: Serviços do *Cluster*

Nome ↑	Status	Tipo	Pods	Namespace	Cluster
postgres	OK	Stateful Set	1/1	wikijs	cluster-wikijs
wikijs	OK	Deployment	1/1	wikijs	cluster-wikijs

Figura 6: *Workload* do Sistema

Para além disso, foi também criada uma *recipe* que torna possível o *undeployment* da aplicação.

4 Tarefa 3

Nesta tarefa, inicialmente abordamos a hipótese da instalação da *stack* ELK. No entanto, acabamos por não a implementar visto que a *GCP* fornece vários tipos de monitorização *built-in*. Assim foi permitida uma simplificação do *deployment* da aplicação. As métricas de monitorização selecionadas desta ferramenta foram a utilização de três componentes chave, o *CPU*, a *RAM* e o armazenamento.

A implementação da *stack* ELK será algo a reter num trabalho futuro, caso tenhamos interesse em dar *deploy* da aplicação num ambiente na qual não tenha serviços de monitorização pré instalados.

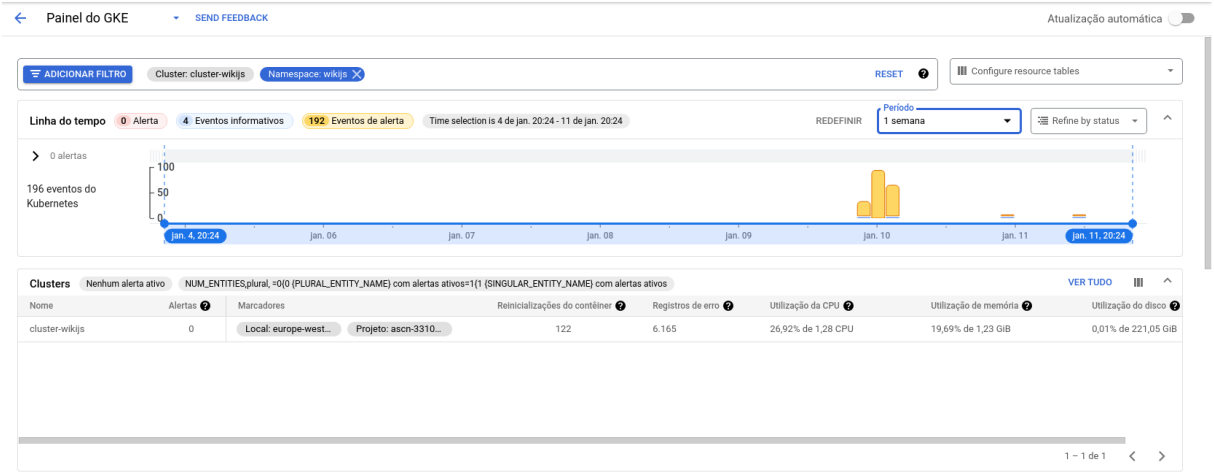

Figura 7: *Dashboard* de Monitorização do *Cluster*

5 Tarefa 4

Nesta tarefa, optamos por utilizar o *JMeter* por ser uma ferramenta de avaliação bastante simples e na qual podemos replicar o comportamento de um utilizador da aplicação. Foram desenvolvidos 3 testes para a avaliação da mesma. Em seguida encontram-se os testes de comportamento de um utilizador simulado pelo *JMeter*:

- **Teste 1:** O utilizador realiza o *login* na sua conta e, em seguida, cria um página com algum conteúdo;
- **Teste 2:** O utilizador realiza o *login* na sua conta e, em seguida, visualiza a página principal;
- **Teste 3:** O utilizador realiza o *login* na sua conta e, em seguida, escreve um comentário na página inicial.

6 Resultados

Em seguida encontram-se os resultados que foram obtidos para os testes referidos na Tarefa 4. Como apenas nos foi permitido correr os testes num utilizador os resultados tal como esperado para este caso, não se revelaram um grande desafio em termos de recursos computacionais.

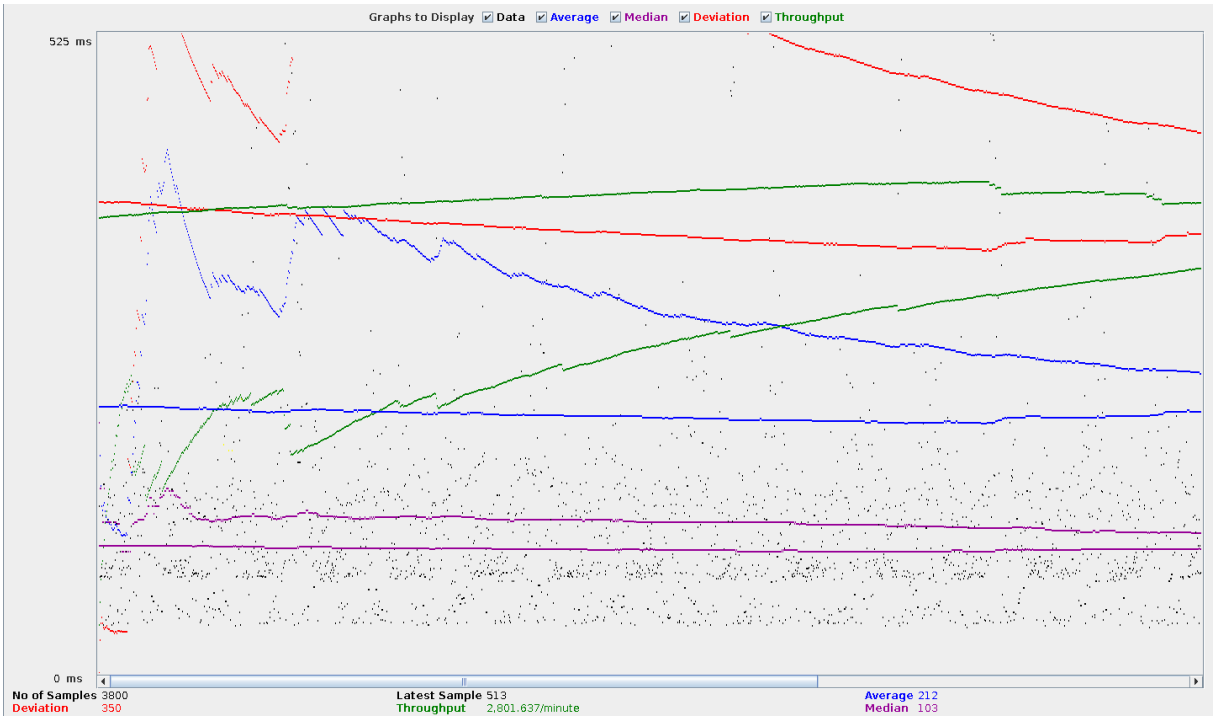

Figura 8: Resultados *JMeter* de Criação de Página

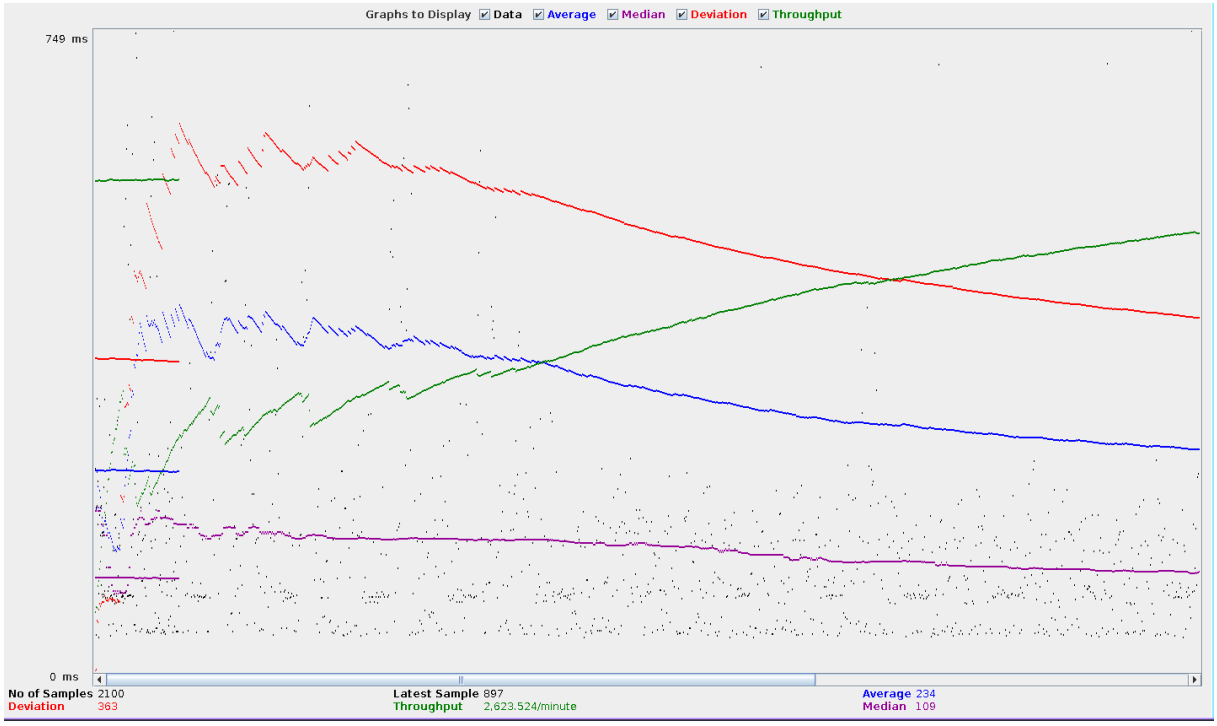

Figura 9: Resultados *JMeter* de *Login* e ver a página principal

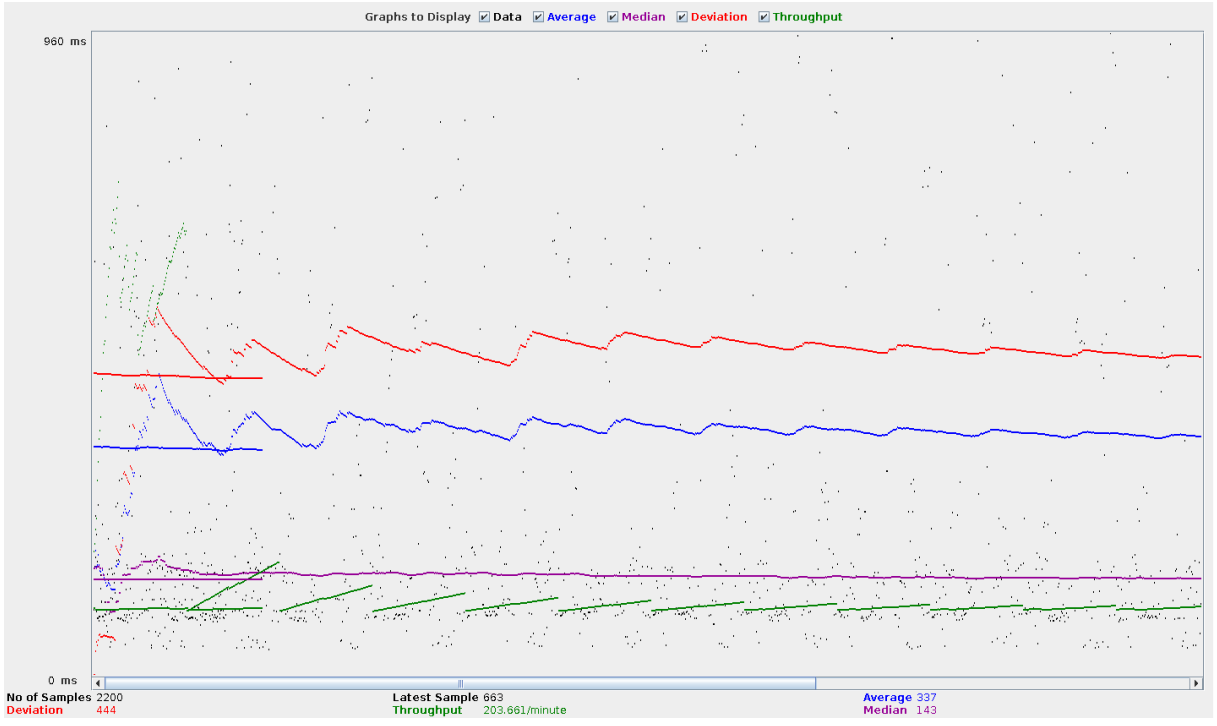

Figura 10: Resultados *JMeter* de escrever um comentário

7 Conclusão e Trabalho Futuro

Dado por concluído o trabalho entendemos que realizamos com sucesso e de forma eficaz as várias tarefas propostas relativamente à instalação, monitorização e avaliação do *Wiki.js*. O projeto desenvolvido foi realizado tarefa a tarefa, tendo sempre sido tomadas as decisões de forma consciente e ponderada.

Algo que poderia ter sido feito de forma diferente e que poderia ser implementado num trabalho futuro, seria a implementação da *stack* ELK, como já tinha sido abordado na Tarefa 3.

Em geral, consideramos que o balanço do trabalho é positivo, as dificuldades sentidas ao longo do desenvolvimento foram superadas, e os requisitos básicos propostos foram cumpridos com sucesso.