Universidade do Minho

14/1/2023

Aplicações e Serviços de Computação em Nuvem

Trabalho Prático

https://github.com/MrZeLee/ASCN_TP

Grupo 5:
a82428 - João Pedro da Costa Gomes
a82174 - Nuno Miguel Freitas de Oliveira
pg47365 - José Luís Moura Marinho

Índice

1 Introdução

Neste relatório, serão utilizados os conhecimentos obtidos na unidade curricular de Aplicacões e Serviços de Computação em Nuvem para implementar e automatizar a instalação da aplicação Ghost assim como a sua monitorização, avaliação e proposta de escalibilidade e resiliência.

O Ghost é uma aplicação open-source de publicação de conteúdo projetada para bloggers e editores online com o intuito de ser uma plataforma simples, elegante e fácil de usar para criar e gerir os seus sites e blogs. O Ghost oferece uma interface limpa e intuitiva para escrever e publicar conteúdo, bem como um sistema flexível de temas que permite que os utilizadores personalizem facilmente a aparência do seu site. O Ghost inclui também SEO incorporado, integração com redes sociais e outros recursos que o tornam uma ferramenta poderosa para a publicação online.

2 Arquitetura da aplicação Ghost

A arquitetura da aplicação Ghost é uma arquitetura baseada em serviços, ou seja, os diferentes serviços fornecidos pela aplicação não estão fortemente ligados uns aos outros e podem funcionar de forma independente.

Esta arquitetura permite um alto grau de flexibilidade, escalabilidade e facilidade de manutenção e facilita também a adição de novos serviços ou atualização de serviços existentes sem comprometer o resto da aplicação.

O front-end da aplicação é geralmente construído utilizando JavaScript e comunica-se com os serviços de back-end através de uma API RESTful. Os serviços de back-end são geralmente construídos usando Node.js e Express.js e um framework de aplicações web para Node.js.

A plataforma Ghost usa também um sistema de gestão de base de dados moderno, como MySQL ou PostgreSQL, para armazenamento dos dados. No nosso caso, foi utilizado MySQL.

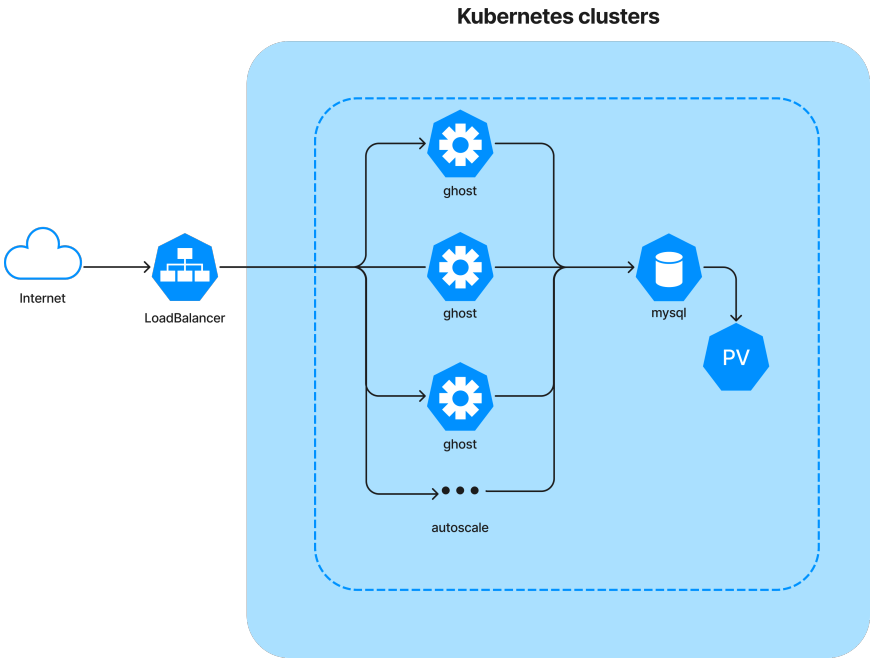

Figura 1: Dashboard de visualização dos recursos utilizados.

3 Desenvolvimento da solução

3.1 Instalação e Configuração Automática da Aplicação

3.1.1 Objetivo

A primeira tarefa consiste em utilizar a ferramenta Ansible para automatizar a instalação e configuração da aplicação Ghost no serviço Google Kubernetes Engine (GKE) da Google Cloud.

3.1.2 Implementação

Para fazer o deployment tiramos proveito das imagens docker do Ghost 5.14.1 e do MySQL 8.0 configurados usando as variáveis do ansible inventory.

Os dados do administrador foram inseridos na base de dados juntamente com os dados no MailGun.

Para a configuração do Ghost foram utilizados diversos comandos:

- Recolha do IP externo criado pelo Load Balancer do GKE (google kubernetes engine).
- Criação de um namespace para melhor divisão dos componentes necessários para a implementação do Ghost. Principalmente usado ficheiros YAML, criados a partir de templates para utilização das variáveis do inventário do ansible.
- Colocação do ghost em espera pela base de dados.
- Reverse engineering da base de dados para colocar o mailgun a funcionar corretamente. (Revelou-se necessário terminar as pods ghost para repovoar a cache).
- Criação do LoadBalencer para expor o deployment do ghost à rede externa.
- Criação PersistantVolumeClaim para que os dados se mantenham em caso de undeploy ou potenciais falhas.
- Implementado Autoscaling das réplicas ghost mediante a métrica do CPU.
- Criação de um readinessProbe com a finalidade de verificar a criação do MySQL dentro do pod.

3.2 Monitorização

3.2.1 Criação de dashboards

De forma a ter uma melhor visualização dos resultados produzidos pelas métricas que estão a ser captadas, foram criadas duas dashboards utilizando as ferraqmentas de monitorização do Google Cloud. Estas dashboards foram divididas numa dashboard dedicada a monitorizar os recursos que estão a ser utilizadas e outra dashboard responsável por monitorizar as operações que estão a ser feitas na rede.

Figura 2: Dashboard de visualização dos recursos utilizados.

Para esta dashboard foram escolhidas métricas que permitem visualizar a carga presente em várias partes do sistema. De forma a visualizar a carga presente no CPU, foi criado um gráfico para monitorizar a utilização, em percentagem, do CPU de cada nodo e também um gráfico que contabiliza o número de processos que está a correr em cada nodo. Foram criados também dois gráficos para visualizar o impacto no disco. O primeiro, mede em percentagem quanta memória se encontra disponível em cada nodo. Já o segundo mede o tempo demorado por acada operação de escrita ou leitura por segundo.

Na nossa opinião, esta dashboard permite ter uma visualização rápida da carga presente no sistema no que diz respeito aos recursos que por este são utilizados.

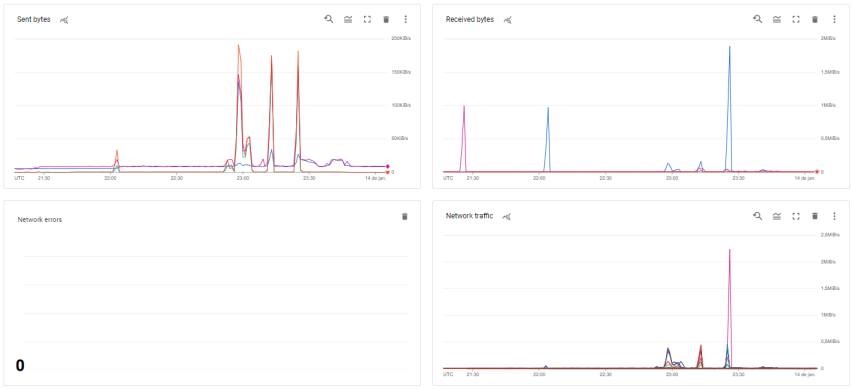

Figura 3: Dashboard de visualização dos recursos da rede.

Para esta dashboard foram escolhidas métricas que permitem visualizar a carga que a aplicação está a aplicar na rede do sistema. Foram criados dois gráficos para monitorizar a quantidade de bytes que está a ser enviada e a quantidade que está a ser recebida para permitir uma visualização do tráfego da aplicação e foi criado também um gráfico que contabiliza a quantidade de erros que estão a ocorrer na rede e um que une os dois anteriores para permitir uma visualização do tráfego total da rede.

Acreditamos que estas métricas sejam as necessárias para conseguirmos perceber o impacto da aplicação na rede.

3.2.2 Criação de políticas de alerta

Uma vez que, num cenário real, é impossível ter uma visualização constante das dashboards, optamos por criar também algumas políticas de alerta. Relativamente às métricas escolhidas, foram criadas três políticas. A primeira, é responsável por criar um alerta caso a carga do CPU de um dos nodos ultrapasse os 80%. A segunda, cria um alerta caso a percentagem de memória disponível num dos nodos desça dos 10%. E por fim, foi criado também um alerta para caso estejam a ser realizadas operações de leitura ou escrita num dos discos que durem mais de 100 milissegundos a cada segundo.

3.3 Avaliação Experimental

De forma a avaliar a performance da aplicação foi utilizado o JMeter com o intuito de testar a resiliência da página a elevadas quantidades de utilizadores. Realizou-se um stress test enviando GET's à pagina inicial fazendo recurso a 300 threads com um ramp-up period de 0.3 segundos.

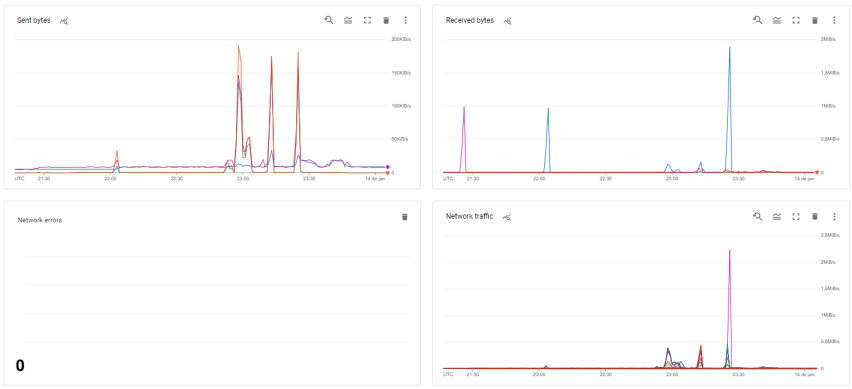

Figura 4: Dashboard de visualização dos recursos utilizados após um teste.

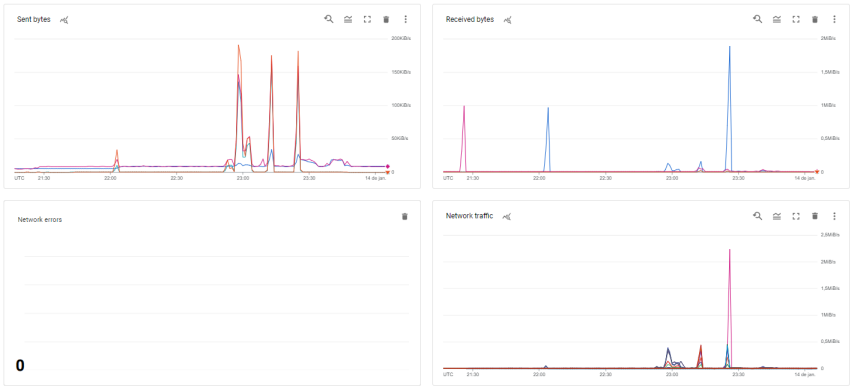

Figura 5: Dashboard de visualização dos recursos da rede após um teste.

Utilizando as ferramentas desenvolvidas na etapa da monitorização, conseguimos visualizar que após a realização deste teste as métricas medidas estavam a atingir valores muito superiores dos normais disparando todos as políticas de alerta previamente criadas.

Na primeira dashboard, conseguimos observar que após o teste a utilização de CPU subiu, a memória disponível diminuiu, o número de processos aumentou e o tempo passado pelo disco em operações de leitura ou escrita subiu.

Já na segunda dashboard, após a realização do teste verificou-se também um aumento no tráfego estando a ser enviados e recebidos mais pacotes do que o habitual. Não houve no entanto erros na rede.

Estes testes foram repetidos com variações nos parâmetros provocando resultados semelhantes em todos os casos.

3.4 Escalabilidade e Resiliência

Com a utilização do Horizontal Pod Autoscaler implementado como uma API de Kubernet e aplicando métricas de utilização de CPU no ghost, possibilitou a capacidade de suportar um número de conexões, geradas por pedidos de utilizadores, superior.

Usando a métrica CPU, colocamos um valor de "targetCPU"de 150m (ou seja 0.150 unidades de CPU). Sendo que quando os pods ultrapassam esse valor são criadas mais pods para suportar os pedidos.

No site da aplicacao ghost, aconselha também a utilizacao de uma CDN (como por exemplo Cloudflare), que mantêm, em cache e distribuido "versoes"da parte "static"do website.

Escalabilidade horizontal a nivel da base de dados não é suportada pela aplicação ghost como referido no site deles.

Neste

4 Conclusão

Em suma, consideramos que o projeto foi concluído com sucesso necessitando de melhorias nalgumas áreas. Relativamente à instalação e configuração automática do Ghost funcionou tudo como pretendido tendo conseguido cumprir com o pedido e automatizar completamente o processo. No que diz respeito à monitorização, consideramos que as métricas que escolhemos são as fundamentais para garantir o bom funcionamento da plataforma mas, possivelmente, iriam surgir novas métricas com o tempo para auxiliar ainda mais o processo. Relativamente à avaliação, poderia ter sido aprofundada criando outros tipos de testes para emular cenários mais realistas e variar ainda mais os parâmetros para ser possível retirar conclusões mais próximas da realidade. Já no que toca à tarefa de escalabilidade e resiliência, achamos que dentro dos recursos disponiveis realizamos um bom trabalho, detetamos a importância de testes automáticos para a verificação destas. Possivelmente com a criação de mais testes e mais especificos conseguiriamos chegar a mais conclusões.

De forma geral, o grupo considera que este projeto teve um papel preponderante no que toca à consolidação da matéria lecionada ao longo do semestre e que este nos possibilitou pôr em prática os conceitos teóricos previamente adquiridos.