

UMinho

Mestrado em Engenharia Informática
Aplicações e Serviços de Computação em
Nuvem (2022/23)

GRUPO 21

Gonçalo Braz (a93178)
Tiago Silva (a93277)
Simão Cunha (a93262)
Gonçalo Pereira (a93168)
Hugo Fernandes (pg50419)



Braga, 20 de dezembro de 2023

Conteúdo

1	Introdução	2
2	Arquitetura e componentes da solução	3
3	Instalação e configuração automática da aplicação	4
3.1	Cluster Kubernetes	4
3.2	Deployment da Aplicação Ghost	4
3.3	Undeployment da Aplicação Ghost	6
4	Tarefas extra	6
4.1	Tarefa 1: Monitorização	6
4.2	Tarefa 2: Avaliação Experimental	7
4.3	Tarefa 3: Escalabilidade e Resiliência	8
5	Resultados	8
6	Conclusões e trabalho futuro	10

1 Introdução

A tarefa proposta para o atual trabalho consistiu na instalação e configuração da aplicação *Ghost*, no serviço **Google Cloud Kubernetes** de forma automatizada através do uso da ferramenta **Ansible**. O propósito do serviço **GKE** é tomar partido do aprovisionamento de clusters de containers que fornecem um ambiente lightweight comparado com as máquinas virtuais normais, com um deployment de aplicações e serviços mais facilitado, assim como ferramentas para monitorização e escalonamento robustos dos mesmos.

O enunciado fornecido pelos docentes dividiu o trabalho total em diferentes tarefas, sendo que para a completude de todas estas, existe uma tarefa base compulsória, que será o principal foco deste relatório, consistindo na instalação da aplicação *Ghost* num cluster supramencionado separando, tanto quanto possível, as diferentes componentes deste em diferentes unidades computacionais.

As restantes tarefas, consideradas extras, serão tratadas na extensão em que foram concluídas num capítulo posterior. Estas conjuntamente consistem na criação de mecanismos de visualização de monitorização do estado do cluster, criação de rotinas para testar de forma automática preferencialmente, o funcionamento da aplicação e o impacto no cluster e a configuração de mecanismos de escalabilidade e resiliência dos containers.

De forma a configurar corretamente a aplicação **Ghost**, seguimos as instruções fornecidas na página de configuração do ghost, juntamente com os diversos guias de instalação, principalmente o do Docker por, tal como no nosso caso, utilizar uma imagem para a instalação, servindo como exemplo para a definição das variáveis de ambiente.

2 Arquitetura e componentes da solução

Para a descrição da arquitetura da solução temos de ter em conta dois fatores, a arquitetura da aplicação que pretendemos provisionar, *Ghost*, que sendo um software já desenvolvido apresenta à partida uma arquitetura já definida e que tivemos de seguir para a sua instalação bem sucedida e, a arquitetura resultante do deployment do *Ghost* no cluster, i.e. os diferentes serviços e pods que são criados no cluster.

A arquitetura do ***Ghost*** segue, num nível macroscópico, a arquitetura comum de servidor cliente, onde o servidor se trata de uma junção da aplicação base *Ghost*, mais propriamente o Ghost-CLI e uma base de dados que tanto pode ser interna ao dispositivo desta primeira instalação, como por exemplo SQLite, ou externa, como MySQL. Esta última é o preferencial para permitir a modularidade do sistema. Olhando com mais algum detalhe, verificamos que o Ghost-CLI fornece, além das ferramentas para responder a pedidos de acesso ao web site *Ghost*, diversas ferramentas de administração para facilitar a manutenção do mesmo. Além disso, de modo a permitir a criação de novos utilizadores de forma natural, i.e. sem acesso de administrador ou à base de dados, é necessário ainda a integração de um serviço de mail, como o utilizado nas aulas práticas, mailtrap.

A arquitetura, como representada pelo próprio proprietário do *Ghost*, pode ser analisado na seguinte ilustração:

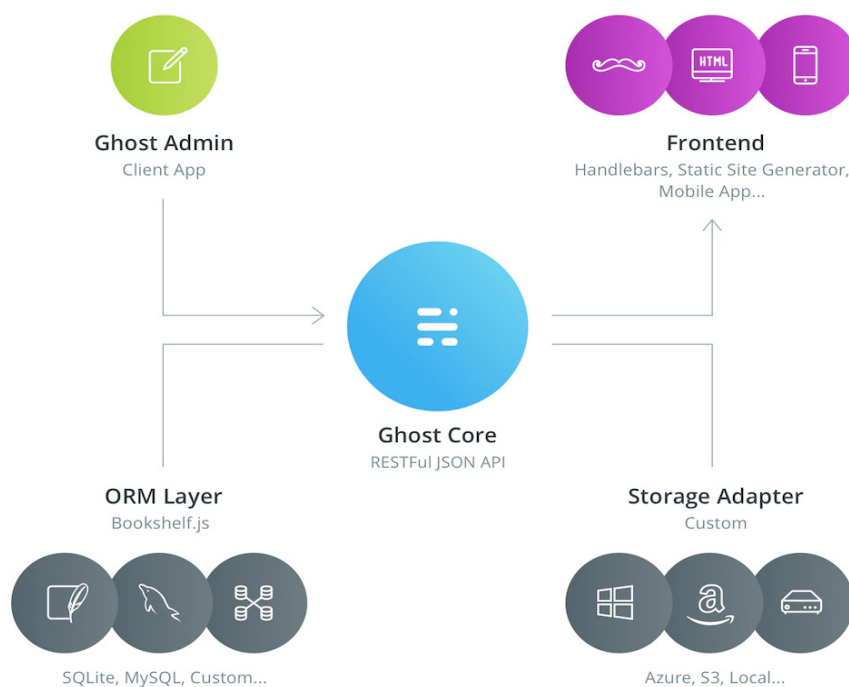


Figura 1: Arquitetura Ghost

O **Cluster** resultante da nossa instalação do *Ghost*, irá modularizar este, dividindo entre dois serviços principais o sistema: o serviço **ghost**, que terá um IP acessível externamente e que é responsável por permitir a todos os utilizadores, inclusive administradores de blogues, aceder ao site do *Ghost* para ver blogues, administrar blogues, criar blogues, registar novo utilizador e dar login; e o serviço de mysql, apenas conectável dentro do próprio cluster, que será responsável por guardar os dados resultantes da utilização do *Ghost*, i.e. informação dos utilizadores, blogs, etc. Os dados guardados, de maneira a serem acessíveis após uma desinstalação do Ghost, terão de ser guardados em algum tipo de armazenamento persistente.

Ambos os serviços são por sua vez compostos por pods que dão deploy nas aplicações que compõem os serviços mencionados. Estes pods poderão aplicar estratégias de escalamento automático e de resiliência a partir da maneira como forem configurados.

3 Instalação e configuração automática da aplicação

3.1 Cluster Kubernetes

Tal como referido anteriormente, será necessário criar na plataforma Google Cloud um cluster Kubernetes através do Google Kubernetes Engine (GKE). Para tal, são utilizados dois *playbooks* - já fornecidos pela equipa docente - o primeiro para a criação e outro para a respetiva destruição do cluster: `create-gke-cluster.yml` e `destroy-gke-cluster.yml`, respetivamente. Estes *playbooks* em específico, tal como requerido, não foram adulterados.

Assim, o cluster será denominado `ascn-cluster` e será uma máquina do tipo `e2-medium`, com 4GB de memória RAM e 4 vCPU's, composto por dois nodos de pods, suficientes para a realização do projeto. Por questões de *billing*, a zona atribuída foi a mais económica, i.e. `us-central1-a`.


<input type="checkbox"/> Status	Name 	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input checked="" type="checkbox"/>	ascn-cluster	us-central1-a	2	4	8 GB	—	⋮

Figura 2: Visualização do cluster Kubernetes no Google Cloud Platform

De forma a automatizarmos os processos inerentes ao trabalho prático, cada um dos elementos do grupo instalou nas respetivas máquinas o `gcloud` e o `Ansible`, maioritariamente através do WSL (Windows Subsystem for Linux).

3.2 Deployment da Aplicação Ghost

A instalação da aplicação *Ghost* é realizada, tal como proposto no código base fornecido, a partir da chamada do *playbook* `deploy-ghost.yml`. Dentro deste, de forma a seguir a arquitetura teremos dois roles que serão executados, `ghost_db` para instalar a base de dados e `ghost_app/deploy-ghost` para instalar a aplicação de client do ghost. De forma a melhorar a organização entre possíveis clusters no GKE, criamos ainda uma role para gerar um `namespace` utilizado por todos os componentes do cluster. O `namespace` é criado utilizando módulos do próprio `k8s`, `kubernetes`.

Serviço de Base de dados

O role de criação do serviço de base de dados é iniciado pela criação de uma de armazenar os dados de forma persistente, mais especificamente, um volume persistente.

Para isto, a primeira tarefa será a de criar uma classe de armazenamento, isto permite distinguir os volumes criados para o propósito da base de dados ou do nosso cluster, de outros já existentes ou que existirão no futuro. Além disso, esta classe define o provedor do volume, que no caso deste projeto, será o `gce-pd`, i.e. google compute engine - persistent disk.

De seguida, de maneira a que um volume persistente seja criado no GKE, é necessário fazer uma *claim* desse volume. Esta *claim* tem de ser configurada de modo a definir a classe a quem devemos pedir o volume (que será a previamente criada), o tamanho do volume e o tipo de acesso do volume, que permite definir como este pode ser acedido e por quem.

Com o volume persistente pronto a ser utilizado, estaríamos prontos a fazer o deployment da base de dados, para a qual nós escolhemos utilizar `mysql`. Para o deployment desta no entanto, necessitamos de fornecer um nome de utilizador e uma palavra passe. De forma a testar a funcionalidade `Secret` do `gce` - utilizada para guardar de forma segura informação

sensível como credenciais -, precisamos antes do deployment de tasks para fazerem download deste secret criado.

Possuindo as credenciais guardadas no secret, atualizamos o template de deployment do **mysql**, alterando os campos de user e password para os lidos e, de seguida corremos este deployment, utilizando o módulo de k8s. Este deployment define que é específico da app **mysql**, de modo a poder ser utilizado para um serviço; a estratégia utilizada para reconstruir os pods, onde definimos como **Recreate**; a imagem que vai ser utilizada, que é sempre a mais recente versão de **mysql**; e o volume a utilizar para guardar os dados e onde, que será o volume criado nas tasks anteriores.

Por último, com o deployment feito, apenas falta expor este serviço, para que possa ser utilizado por outros serviços no cluster. Novamente utilizando o módulo do k8s, criamos um serviço que procura o deployment **mysql**, de tipo **ClusterIP**, ou seja, apenas acessível dentro do cluster, e que utiliza a port 3306.

Serviço Ghost

A criação do serviço **Ghost** é iniciado, no mesmo sentido que a task acima mencionada, com a atualização do template de deployment do ghost, com as credenciais da base de dados.

De seguida fazemos deploy do ghost onde de forma semelhante ao deploy supramencionado, temos de definir que é destinado à app ghost; utiliza a estratégia **recreate**; pode ter no máximo 1 réplica, isto pois na documentação de ghost isto é o recomendado; e utiliza a imagem do ghost, na versão 5.14.1, como sugerido pela equipa docente.

Para a instalação da imagem do ghost, criamos um conjunto de variáveis ambiente, que permitem a configuração de diversas funcionalidades na aplicação. Estas são: a base de dados a que se deve conectar, com respetivo user e password e o serviço que é sua host; o url do ghost, que terá de ser atualizado mais tarde pois ainda não conhecemos o IP do serviço ghost, visto que este não foi criado; e um conjunto de variáveis para definir um serviço de mail que permite a inscrição de novos utilizadores no ghost, que envolvem, o tipo de transporte de mail, o serviço de mail, o host, a porta para o aceder, as opções de segurança e as credenciais de autenticação do serviço.

Prosseguimos com a exposição do serviço ghost, especificando que ele irá juntar os pods que compõe a app ghost, e com type **LoadBalancer**, de forma a expor o serviço externamente, i.e. para ser acedido fora do cluster, na porta 80, para ser acessível também na eduroam.

De seguida, o próximo passo consiste na criação/atualização de um administrador do ghost, inserindo diretamente na base de dados um utilizador predefinido pelos docentes, com nome **ascn**, password **ascn123** e mail **ascn@example.com**. No entanto, antes de fazer isto, temos de confirmar que a tabela de user já existe, pois pode ainda estar no processo de ser criada. Para isto, procuramos um pod que tenha dado deploy no **mysql** e passamos um comando a este usando o módulo **k8s_exec** para verificar se já foi criada a tabela.

Como anteriormente mencionado, é necessário atualizar o url do ghost, com o IP atribuído ao serviço exposto. Para isso, utilizamos o módulo **k8s_info** que, no resultado da procura pelo serviço, se este existir, terá informação sobre o seu IP. Com esta informação, atualizamos o ficheiro de deployment do ghost e damos deploy deste novamente.

Por último, sendo este passo já uma tarefa extra, verificamos se na conta de google cloud utilizada existe alguma dashboard com nome **Ghost-Monitor** e, se não existir criamo-la.

Após estes passos, o ghost já deverá ser acessível por um browser, utilizando a url:

`https://<<ghost_service_ip>>:80`

3.3 Undeployment da Aplicação Ghost

Além de ser possível instalar o serviço Ghost, também nos foi instruído que criássemos uma automatização para o desinstalar, com a opção de manter ou não os dados persistentes do sistema. Esta opção é inibida através do uso da flag `delete_data=true`.

O primeiro passo da desinstalação consiste na destruição dos serviços, pois se destruíssemos primeiro os deployments, como os serviços ainda estariam ativos, eles irão repô-los.

Após os serviços serem destruídos, fazemos o mesmo com os deployments.

Por último verificamos se a flag `delete_data` está definida e, se tal, com valor `true`, e em caso positivo apagamos o `PersistentVolumeClaim` criado durante o deployment do `mysql`.

Caso esta flag não esteja definida ou tenha o valor de `false`, então este volume persistente não será apagado, ou seja, num novo deploy do ghost, todos os users, blogs e outros dados que se encontravam no deploy anterior já irão estar disponíveis à partida neste.

Chegado a este ponto, já é possível criar e aceder à aplicação ghost, assim como a desinstalar, seguindo os requisitos solicitados pela equipa docente, podendo considerar então, que a etapa base do projeto está realizada.

4 Tarefas extra

Como formas de explorar melhor as funcionalidades das kubernetes, assim como da google cloud, foram propostas no enunciado tarefas extra para serem realizadas, que aumentam o aproveitamento do trabalho.

Estas englobam a exploração de ferramentas de monitorização do estado do cluster e dos serviços criados, testagem dos mesmos e exploração de configurações do deployment de modo a aumentar a escalabilidade e resiliência do sistema.

4.1 Tarefa 1: Monitorização

Nesta primeira tarefa, o processo seguido foi a exploração das ferramentas de monitorização oferecidas pela google cloud, isto é as Dashboards, e criar uma dashboard personalizada focada na monitorização deste projeto.

Dentro da customização de uma dashboard tivemos de fazer escolhas sobre quais os recursos que pretendíamos observar. Os recursos escolhidos para observação foram:

- Utilização média de CPU do cluster
- Pedidos de utilização média de CPU por serviço
- Número recomendado de requests de core por replica: de modo a, em caso de haver escalabilidade perceber como está a ser balanceado o uso de CPU
- Número de bytes recebidos por app: representante do número de pedidos feito por clientes a aceder ao ghost e do ghost para o mysql
- Número de bytes enviados por app: representante do número de respostas feito pelo ghost aos clientes e da mysql ao ghost
- Volume usado por pod: permite analisar a quantidade de volume utilizado por exemplo, do volume persistente
- Latências das respostas de cada app: permite perceber como o desempenho dos serviços estão a variar em tempos de repouso e em tempos de teste
- Quantidade de armazenamento efêmero usado por app: permitiria, caso necessário, perceber se seriam necessários volumes persistentes para o deploy do ghost

A dashboard construída tem o seguinte aspeto:



Figura 3: Dashboard de observação do cluster

4.2 Tarefa 2: Avaliação Experimental

Nesta tarefa, o objetivo foi criar um método de validação da correta execução da tarefa base. Para isto, além de verificar que a execução do playbook `test-all.yml` fornecido pelos professores era concluído com sucesso, significando que o deployment e undeployment do ghost, à partida, funcionam como devido, criamos uma série de testes extra para verificar que as funcionalidades do ghost se encontram realmente disponíveis.

Assim como aprendemos nas aulas práticas, fizemos testes automáticos, com ajuda da ferramenta do Apache JMeter, como: GET da página de login do admin, login do admin, create post e sign-up de um utilizador (neste teste usamos `__RandomString` nos nomes e emails dos utilizadores). No auxílio da criação destes testes utilizamos uma ferramenta chamada Blazemeter que guarda o tráfego HTTP da página e consegue converter num teste do JMeter. Usamos esta ferramenta como uma extensão do Google Chrome.

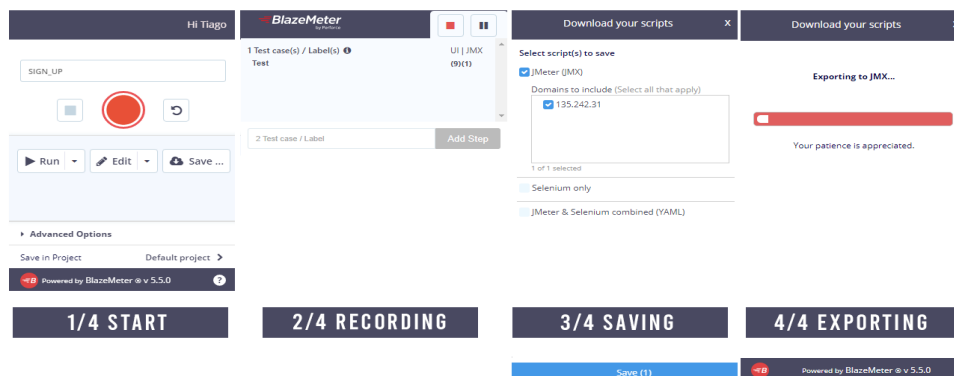


Figura 4: Criando um teste com auxílio do BlazeMeter

Com os testes criados para jMeter, criamos templates semelhantes mas com chamadas de variáveis yaml no lugar do IP e número de threads, de modo a poderem ser substituídos os testes

automaticamente através da execução do playbook `test-all.yml`. Neste realizamos os vários testes de forma automática e numa só execução, sendo facilmente replicável, experimentando os testes com diferentes números de threads.

4.3 Tarefa 3: Escalabilidade e Resiliência

Nesta última tarefa, o objetivo era de atualizar a configuração da tarefa base de modo que, os deployments fossem capazes de escalar automaticamente dependendo do número de pedidos recebidos. Além disso, para aumentar a resiliência poderiam ser obrigadas a existir sempre duas réplicas da base de dados de modo que se uma falha-se ainda teríamos outra disponível.

No entanto, o nosso grupo não foi capaz de explorar esta tarefa.

5 Resultados

Em relação ao trabalho base, numa experimentação do deployment verificamos que cumprimos os resultados pretendidos desta fase, i.e. o site ghost fica acessível num browser, permitindo aceder à página de admin através do utilizador fornecido no enunciado; é possível criar novos blogs; subscrever ao blogue como novo utilizador.

Da mesma forma, o undeployment segue a proposta do trabalho, sendo que todos os serviços e deployments são destruídos quando o playbook é corrido, mas os dados persistentes apenas são apagados quando passada a flag `delete_data` com valor `true`.

Na secção 4.2 das tarefas consideradas como extra neste trabalho prático, criámos vários testes para GET da página de login do admin, login do admin, create post e sign-up de um utilizador - este último criando utilizadores com nome e mails aleatoriamente criados-, através do uso da ferramenta **Apache JMeter**, uma vez que acreditámos que são esses os pontos na aplicação cujo um alto tráfego pode influenciar no seu desempenho.

Seguem-se os resultados da execução destes testes com aumento sequencial de número de threads concorrentes a executar cada teste, 1, 10, 100 e 1000 threads.

Teste	Nº Threads	RT médio (ms)	Throughput(s)	Erro (%)
Aceder página de login	1	392	2.6	0
Aceder página de login	10	397	7.7	0
Aceder página de login	100	486	45.2	0
Aceder página de login	1000	6227	118.5	0
Login admin	1	2748	0.207	0
Login admin	10	1677	2.1	0
Login admin	100	10651	6.6	29
Login admin	1000	39059	20.9	83.2
create post	1	1108	0.578	0
create post	10	1374	3.4	70
create post	100	10390	7.1	92
create post	1000	56948	8.8	94.5
sign up	1	959	1	0
sign up	10	726	6.3	50
sign up	100	1718	31.9	95
sign up	1000	11478	55.9	99

Tabela 1: Tabela de resultados

Antes de analisar os dados, é importante notar que tanto o ghost como o serviço de mail utilizado, mailtrap, apresentam um limite de uso, aumentando ainda mais a % de erro quanto

maior o número de threads. Este é 100 login por IP no caso do Ghost e de 100 emails por inbox no mailtrap.

Analisando agora os resultados obtidos, verificamos que para operações simples, como aceder à página, a resposta foi de 100%, porém mesmo assim houve um acréscimo notável na quantidade de tempo médio de resposta.

No entanto, quanto mais computacionalmente complicado a tarefa, mais rapidamente sobe a taxa de erro, como facilmente se nota para o teste de criação de um post e de criação de um utilizador.

No caso do login do admin, seria impossível, devido à limitação imposta mencionada, o login do mesmo user mais do que 100 vezes seguida, mas mesmo assim, no caso de 100 threads, em que ainda seria possível todas terem sucesso, a taxa de erro é relativamente grande.

Podemos concluir então que seria necessário uma distribuição do servidor que trata de receber e responder aos pedidos vindos da frontend, no entanto, limitações de APIs externas impõe um bottleneck que não é otimizável por nós.

Observando ainda a dashboard criada, verificamos que a quantidade de espaço alocada para a base de dados é mais do que suficiente para a quantidade de trabalho testada, porém a quantidade de CPU do cluster chega em tempos quase a valores máximos.

6 Conclusões e trabalho futuro

Fazendo uma reflexão sobre o trabalho realizado, compreendemos que conseguimos explorar as ferramentas kubernetes e google cloud de forma a realizar o trabalho base proposto de forma clara e automatizada, tomando proveito de módulos de ansible que facilitam o manuseamento destas. Ainda mais, embora não tenhamos conseguido realizar todas as tarefas extra, mais especificamente a garantia de escalabilidade e resiliência, cumprimos a exploração de ferramentas de monitorização de forma a adequar-se a análises úteis do cluster e serviços criados, e criámos um conjunto de testes que exploram diferentes áreas do site ghost além de um simples pedido de acesso à página inicial deste, como por exemplo a criação de users e posts, que criam um impacto não só no serviço ghost mas também na base de dados.

Consideramos então que obtivemos experiência nestas ferramentas que permitem a distribuição de sistemas e que são prática do mercado, ainda tendo como trabalho futuro a possibilidade de estudar as mecânicas de escalonamento e resiliência que omitimos no projeto.