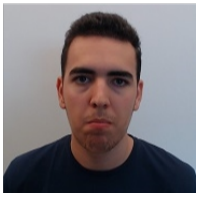
UMinho - Mestrado Engenharia Informática

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

Automatizar o processo de instalação, configuração, monitorização e avaliação da aplicação Ghost

André Martins
PG50224

João Moreira
PG50465



João Giesteira
PG50476

Lídia Sousa
PG50551

Pedro Ferreira
PG50695

Repositório URL: `https://github.com/Noodle129/ASCN`

Braga, 14 de janeiro de 2023

Conteúdo

Lista de Figuras

1. Introdução

A realização deste trabalho, no âmbito da UC de Aplicações e Serviços de Computação em Nuvem, foi de encontro ao enunciado apresentado pela equipa docente que tinha como objetivo: **"exercitar os conhecimentos adquiridos na cadeira de forma a automatizar o processo de instalação, configuração, monitorização e avaliação da aplicação Ghost"**.

Ao longo deste documento iremos apresentar as várias etapas do desenvolvimento da solução implementada.

2. Ghost

O Ghost é uma aplicação poderosa para os novos criadores de media publicarem, partilharem, e desenvolverem um negócio em torno do seu conteúdo. Vem com ferramentas modernas para construir um *website*, publicar conteúdos, enviar *newsletters* e oferecer subscrições pagas aos membros.

2.1 Arquitetura da solução

A solução implementada pode ser descrita pela figura abaixo apresentada.

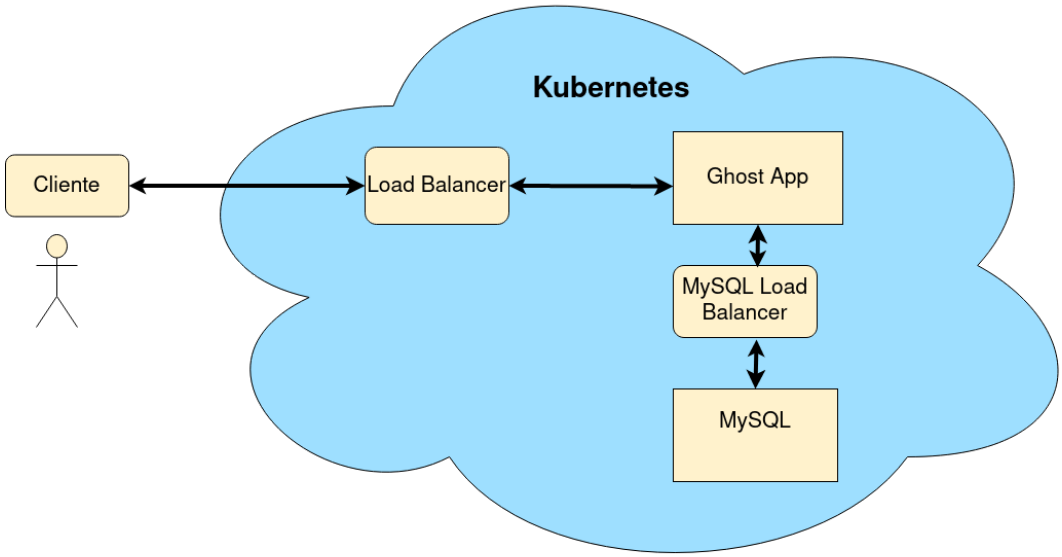


Figura 2.1: Arquitetura

Em suma, a aplicação do Ghost está exposta num LoadBalancer pelo qual o utilizador acede. Relativamente à conexão com a base de dados, a aplicação do ghost comunica com o LoadBalancer onde o MySQL está exposto.

3. Automatização

3.1 Playbooks

Para a automatização dos diferentes processos, tiramos proveito da ferramenta *Ansible*, através da utilização de *playbooks*. Os *playbooks* são um conjunto de instruções que especificam como um determinado sistema deve ser configurado, podem ser usados para automatizar tarefas simples ou complexas e podem ser facilmente compartilhados entre equipes de operações. A automatização foi realizada também com o suporte das ferramentas GKE (Google Kubernetes Engine) e GCP (Google Cloud Platform), tanto na realização do *deployment/undeployment* da aplicação Ghost, como na parte da monitorização.

Os processos de automatização pode ser enumerados paralelamente com a enumeração dos *playbooks*. Os *playbooks* disponíveis são os seguintes:

- Create GKE Cluster
- Destroy GKE Cluster
- Deploy Ghost
- Undeploy Ghost
- Test All
- Test 1

3.1.1 Create/Destroy GKE Cluster

Os dois *playbooks* em questão foram disponibilizados pela equipa docente e não foram alterados. Neles estão instruções para criar/destruir os *clusters*.

3.1.2 Deploy Ghost

Este *playbook* está dividido em 3 partes:

- Configuração e Instalação da Aplicação Ghost
- Atualizar a base de Dados com o administrador de Blog
- Configuração e Criação da *Dashboard* de Monitorização

Neste *playbook* optamos por implementar *tags* em cada uma das partes acima referidas para ao executar o *playbook* ser possível utilizar a *flag* '–tags' para executar certas *tags* específicas, ou a *flag* '–skip-tags' para executar o *playbook* de modo a dar *skip* a certas 'tags'.

Configuração e Instalação da Aplicação Ghost

De modo a ter os componentes da aplicação Ghost a executar em *pods* distintas, primeiramente temos de lançar a base de dados que o Ghost irá utilizar. Para tal temos 3 ficheiros: o **deployment**, o **PVC** e o **service**. O ficheiro do PVC permite a persistência dos dados, nele cria-se o 'persistent volume claim' onde a base de dados irá guardar os seus dados. O ficheiro do *deployment* aloca o *pod* onde a base de dados irá correr, e é onde especificamos a imagem do *mysql* e as suas respetivas variáveis de ambiente. É também aqui onde se faz o *claim* do PVC anteriormente criado. Por fim, o ficheiro do *service* é onde expomos o serviço lançado anteriormente na porta 3306. Todos estes ficheiros são fixos, exceto o ficheiro '*db-deployment.yml*' que, para garantirmos a modularidade, usa as variáveis do ficheiro de *inventory*('*gcp.yml*'). Para aplicar as variáveis a esse ficheiro usamos o módulo 'template' do *Ansible*.

Estando a base de dados operacional procedemos então à instalação do Ghost. Como este não armazena nenhuns dados que necessitem de ser persistidos, não é necessário alocar um PVC. Resta assim apenas criar o *deployment* e o *service* para o Ghost. A ordem da execução destes dois componentes é crucial, pois para não termos de alterar as configurações do Ghost depois de ele já estar ativo e dar *restart* no mesmo, optamos por criar o *service* primeiro para saber em que endereço IP o Ghost irá ser exposto. De realçar que neste serviço fazemos um *forwarding* das portas. O *service* expõe a porta 80, para ser acedido pela mesma, mas encaminha todo o tráfego para a porta 2368 que é a porta *default* do Ghost. O ficheiro do *service* é fixo.

Depois de termos o IP onde o o serviço do Ghost estará exposto podemos atualizar essa informação no ficheiro de *deployment* do Ghost através do módulo 'template', e ao mesmo tempo também são atualizadas outras variáveis do projeto. O ficheiro do *deployment* possui a imagem do Ghost que é utilizada (bem como a sua versão) e várias variáveis para a configuração da aplicação (mais abordadas posteriormente).

Atualizar a base de Dados com o administrador de Blog

De modo a criar automaticamente um administrador de Blog utilizamos a aplicação do *mysql*, através do *pod* onde a mesma está a correr, para adicionar diretamente à base de dados os dados do administrador de blog.

Configuração e Criação da *Dashboard* de Monitorização

É criada uma nova nova *dashboard* no caso de não existir alguma já criada, através de um ficheiro *json* de configuração da *dashboard*. Para o ficheiro estar de acordo com as especificações do projeto, usamos o módulo *template* para substituir as variáveis que correspondem ao *ID* do projeto, e ao nome dos *pods* do Ghost e *mysql*. Utilizamos então o ficheiro resultante da operação anteriormente referida para criar a *dashboard* na *GCE*.

3.1.3 Undeploy Ghost

Assim como no Deploy Ghost, este *playbook* está também dividido em 3 partes:

- Apagar todos os *pods*, serviços e *deployments* do Ghost e da base de dados
- Apagar a *Dashboard*
- Apagar o PVC da base de dados

Desinstalação da Aplicação Ghost

Para proceder à desinstalação do Ghost procedemos à remoção dos *pods*, dos *services* e dos *deployments* do Ghost e da base de dados. Para selecionar os componentes corretos tiramos

partido das *labels* que utilizamos na criação das componentes, mais concretamente para a base de dados usamos a *label* 'app=mysql' e para o Ghost utilizamos a *label* 'app=ghost'. No fim de eliminar as componentes em questão, esperamos até que as mesmas acabem o seu processo de remoção.

3.1.4 Test All

Este *playbook* foi também disponibilizado pela equipa docente e não foi alterado, nele são importados os *playbooks* por nós criados ('deploy-ghost' e 'undeploy-ghost') e é utilizada a role 'test_ghost', criada também pela equipa docente, que efetua um pedido HTTP ao site do Ghost e verifica se o site está operacional.

3.1.5 Test 1

Este playbook foi criado para, através do *Ansible*, conseguirmos avaliar experimentalmente a instalação e configuração da aplicação Ghost. Nele, podemos evidenciar os seguintes tópicos:

- Criação de uma *dashboard*
- Remoção de uma *dashboard*
- Persistência dos dados
- Remoção dos dados Persistentes
- Site Funcional

4. Ansible-Vault

No decorrer deste projeto foram utilizadas várias passwords(databse, mailtrap,...) pelo que estas não poderiam estar em texto corrente. Através do 'ansible-vault' encriptamos um ficheiro de texto que possuía diversas variáveis com as passwords que não queríamos que fossem reveladas. Esse ficheiro foi passado como um ficheiro de variáveis a cada role que necessitaria de ter acesso a uma password que estava encriptada.

```
vars_files:
  - secretFile.yml
}
```

Devido à utilização do *ansible-vault* é necessário fornecer a password quando se corre o *playbook*, por exemplo, com a *flag* '–ask-vault-pass'. No desenvolver da solução, para simplificar, utilizamos a *flag* '–vault-password-file=password.txt', onde no ficheiro 'password.txt' temos a password para desencriptar o ficheiro com as passwords.

5. Exploração Ghost

5.1 Variáveis de ambiente

Após a exploração do site do Ghost, percebemos que seria necessário inserir o URL do site na configuração do Ghost, pois para os reencaminhamentos do próprio site, éramos reencaminhados para o URL 'localhost:2368' ao invés de '<IP_Ghost>:80'.

```
- name: url
    value: http://{{ ghost_ip }}:{{ ghost_expose_port }}/
}
```

De modo a ser possível dar *login*, subscrever ao *newsletter*, dar *sign-in*, entre outros, foi necessário configurar a parte do email. Embora o serviço de email esteja configurado com o MailTrap, os emails não são enviados para a *inbox* dos email em questão, mas sim para a *inbox* do Mailtrap da conta criada para esse efeito. Fazer o *forwarding* desses emails seria possível mas implicaria custos monetários, pelo que não foi efetuado.

```
- name: mail__options__service
    value: Mailtrap
- name: mail__transport
    value: SMTP
- name: mail__options__host
    value: smtp.mailtrap.io
- name: mail__options__auth__user
    value: {{ user_mailtrap }}
- name: mail__options__auth__pass
    value: {{ password_mailtrap }}
- name: mail__options__port
    value: "2525"
- name: mail__from
    value: "Admin Grupo29 <{{ mail }}>"
```

6. Monitorização

O processo de *deployment* do *Ghost* e tudo mais que envolve o uso de *kubernetes* gera bastante informação útil, informação essa que ajuda na gestão pro ativa dos *clusters* de *kubernetes*. Assim, para poder visualizar e analisar essa informação, é necessário explorar o processo de monitorização.

Para isto, usamos a ferramenta de monitorização disponibilizada pela plataforma Google Cloud juntamente com os *playbooks* de *Ansible* para auxiliar o processo de automatização, de forma a criar uma *Dashboard* com as métricas que consideramos relevantes para a monitorização, chamada *Ghost_Monitoring*.

6.1 Gráficos

6.1.1 *Nodes, Pods and Containers*

Através de métricas que monitorizam estes componentes, agrupamos todas pelo critério *"count"* que conta o número de instâncias em que essas métricas são testadas, dando assim todos os números de cada componente.

6.1.2 *Logging throughput of Nodes*

Volume médio de *log bytes* gerado nos nodos por *user* e *system workloads* nos Nodos. Isto permite-nos ter uma noção do tráfego gerado nos nodos, e como tal, da carga de trabalho a que cada nodo possa estar sujeito. Escolhemos esta métrica pois achamos-a importante para conseguir estimar a quantidade de armazenamento necessário para guardar *logs* em determinados períodos de tempo, e identificar se os nodos estão com sobrecarga.

6.1.3 *Total memory*

Neste gráfico analisámos 3 métricas de memória dos nodos. Estas são a memória total disponível para os Nodos, a quantidade de memória alocável para os nodos e a memória dos nodos em uso. Para cada uma destas usamos o percentil 50 dos valores entre todos dos nodos de forma a obter valores baseados na mediana de cada nodo, sendo esta uma estatística forte contra *outliers*. Isto permite-nos ter uma ideia bastante completa do estado da memória dos Nodos, e tomar decisões sobre o escalonamento do armazenamento.

6.1.4 *RTT Latencies per GKE node*

Usamos uma distribuição de RTT medida sobre conexões TCP a partir do nó GKE para os pontos finais na nuvem, incluindo a partir de *Pods* dentro do nó, sobre vários percentis (5º, 50º e 95º). Saber os RTTs é importante pois dá-nos informações sobre latência e congestionamento da rede o que poderá ser útil para resolver problemas com a mesma.

6.1.5 *Memory usage in Containers by pod name*

Aqui temos a quantidade de Bytes usados de memória média por cada *container*, filtrando estes pelos nomes dos *pods* que são criados no *deployment*, usando *regex* para isto. Estes nomes são carregados para a *dashboard* usando um *template* no *Ansible*. Achamos esta métrica extremamente relevante pois permite identificar possíveis problemas de recursos antes que eles sejam críticos. Temos ainda para cada *container* a distinção de *evictable* e *non-evictable memory*, o que é importante pois se um *container* estiver a usar muita memória *non-evictable*, isso pode indicar um vazamento de memória ou um problema de configuração de limites de memória. Monitorizar essas informações também ajuda a garantir que os *containers* tenham recursos suficientes para executar seus trabalhos de forma eficiente. Filtramos estes *containers* por nome dos *pods* para distinguir os que estão a correr com o *ghost* dos com *mysql* e para futura escalabilidade caso corrêssemos mais do que um *pod* por *container*.

6.1.6 *Containers CPU limit utilization*

Este gráfico apresenta a fração do limite de CPU que está a ser usada atualmente no *container*, entre 0 e 1, sobre vários percentis (5º, 50º e 95º). Usamos isto de forma a detetar problemas antes que se tornem críticos, por exemplo, sobrecarga do mesmo, assim como garantir que os recursos sejam suficientes para que os *containers* possam executar as suas funções eficientemente e sem interrupções.

6.1.7 *Container starts and restarts (total)*

Analisamos o número de *starts* e *restarts* dos *containers* pois, se estes estão a ser recomeçados frequentemente, pode ser sinal de que algo está a causar erros ou falhas. Para além disso, monitorizar o número de *starts* e *restarts* pode também ajudar a identificar problemas de escalabilidade, já que *containers* que são reiniciados frequentemente podem indicar que a capacidade do sistema está a ser excedida.

6.1.8 *Containers Ephemeral storage usage*

Aqui temos representado o uso de *ephermeral storage* nos *containers* sobre vários percentis (5º, 50º e 95º). Escolhemos este gráfico porque o *ephemeral storage* pode ajudar a identificar problemas de capacidade e desempenho. Se o armazenamento estiver a ficar cheio, pode causar problemas como lentidão, falhas e instabilidade no *container*.

6.1.9 *Pods Bytes received*

Analisamos o número de bytes recebidos pelos *Pods* sobre vários percentis (5º, 50º e 95º). Consideramos isto relevante porque pode indicar o uso da rede e o tráfego de entrada e isso pode ser útil para identificar problemas de desempenho, como lentidão, falhas na comunicação entre pods e problemas de escalabilidade, como a necessidade de adicionar mais recursos de rede. Além disso, monitorizar o número de bytes recebidos também pode ajudar a detetar problemas de segurança, como ataques de negação de serviço (DoS) ou acesso não autorizado.

6.1.10 *Pod Bytes transmitted*

O número de bytes transmitidos pelos *Pods* é importante por razões muito parecidas ao anterior e ainda para detetar vazamento de dados sensíveis. monitorizar o número de *bytes* transmitidos é importante para garantir o desempenho, estabilidade e segurança dos sistemas. Esta análise foi também efetuada sobre vários percentis (5º, 50º e 95º).

Nota: Como referido, usamos em vários gráficos uma análise sobre 3 percentis diferentes (5º, 50º e 95º). O percentil 50 representa a mediana e como tal é uma boa medida para considerar uma generalidade. O percentil 95 ajuda a ter uma noção de casos extremos. Analisar vários percentis diferentes é importante para obter uma visão completa e detalhada do desempenho e distribuição dos dados, o que ajuda a identificar problemas de desempenho, tendências e padrões, e ajuda a planear a escalabilidade.

Figura 6.1: Dashboard '*Ghost_Monitoring*'

7. Testes Automáticos

7.1 Avaliação Experimental

Com o *playbook* 'test1' é possível avaliar a instalação da aplicação Ghost.

7.1.1 Site Funcional

Chamar a role do 'test_ghost':

```
TASK [ghost_deploy : Wait for Ghost Pod to be ready] ************************************
changed: [localhost]

TASK [test_ghost : Refresh inventory] ***************************************************

TASK [test_ghost : Check that you can connect (GET) to Ghost and it returns a status 200] *
ok: [localhost]
```

Figura 7.1: Teste Site Ghost

7.1.2 Criação da Dashboard

Verificamos que não existe a dashboard:

```
TASK [test_1 : [1]Check Dashboard] ********************************
changed: [localhost]

TASK [test_1 : [2]Display Dashboard Info (nao existe)] ************
ok: [localhost] => {
    "msg": []
}
```

Figura 7.2: Não existe Dashboard

Chamar a role 'dashboard':

```
TASK [dashboard : Check Dashboards] ******************************
changed: [localhost]

TASK [dashboard : Update Dashboard File] *************************
changed: [localhost]

TASK [dashboard : Create 'Ghost_Monitoring' Dashboard] ***********
changed: [localhost]
```

Figura 7.3: Criar Dashboard

Verificamos que já aparece a *dashboard*:

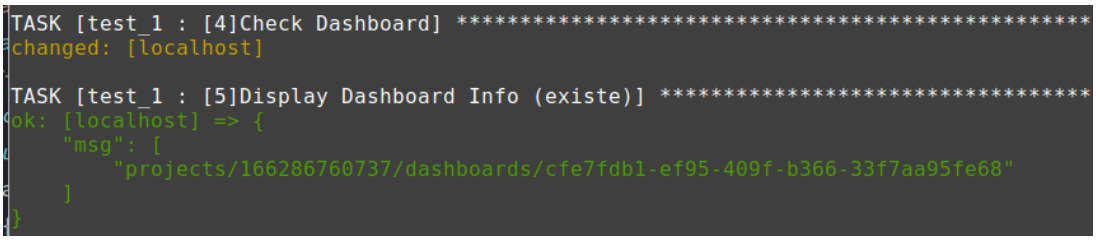


Figura 7.4: Existe Dashboard

7.1.3 Remoção da Dashboard

Sabendo que existe *dashboard*:

Figura 7.5: Existe Dashboard

Apagou-se a *dashboard*:

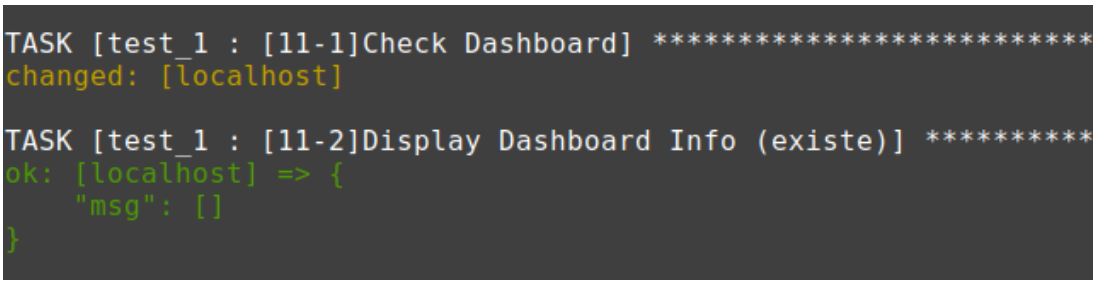


Figura 7.6: Não existe Dashboard

Dashboard já não existe:

7.1.4 Persistência dos dados

Sabendo que não existe dados na tabela 'members':

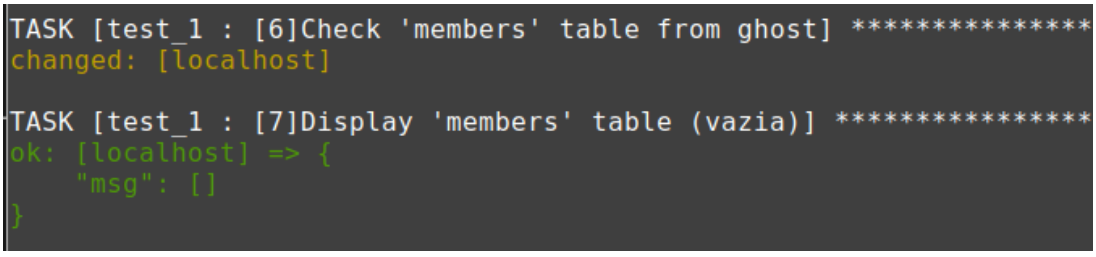


Figura 7.7: Tabela 'members' está vazia

Ao adicionar um membro:

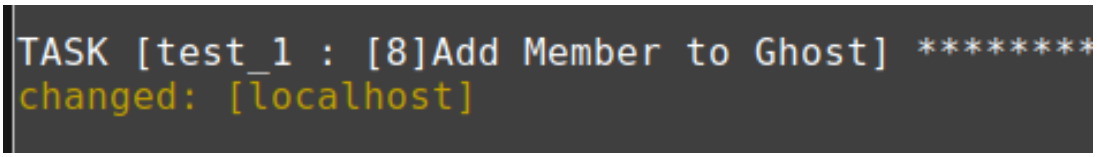


Figura 7.8: Adicionar um membro

A tabela 'members' passa a ter dados:

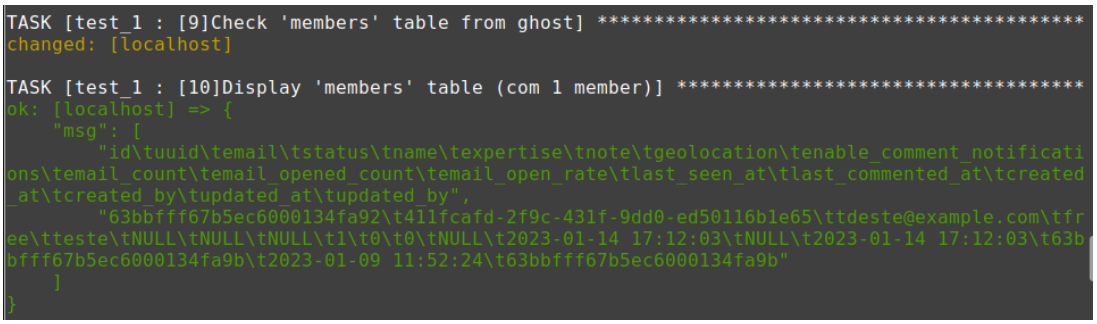


Figura 7.9: Adicionar membro

Após se ter realizado o undeploy_ghost e depois o deploy_ghost podemos ver que os dados não se perderam:

Figura 7.10: Dados Persistiram

7.1.5 Remoção Dados Persistentes

Sabendo que existem dados

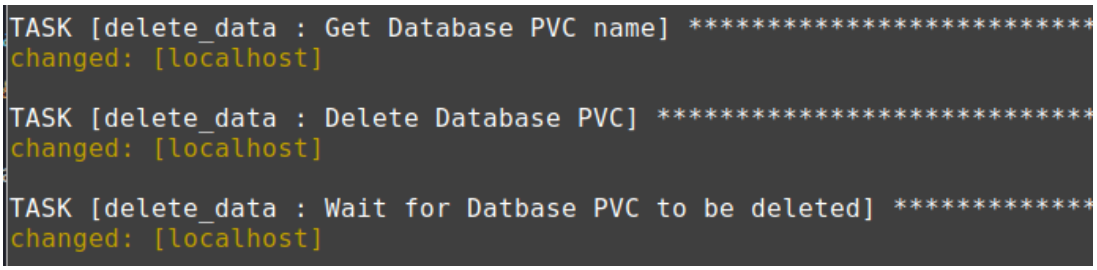


Figura 7.11: Dados tabela 'members'

Após fazer um undeploy_ghost com o delete_data, podemos ver que os dados desapareceram:

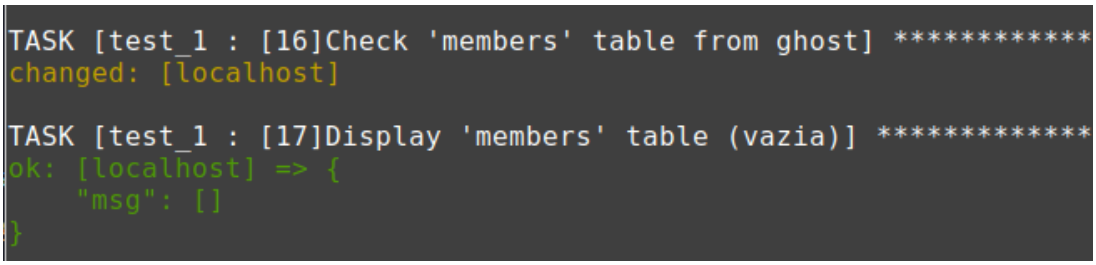


Figura 7.12: Tabela 'members' vazia

8. Conclusão

Tendo tudo previamente referido em conta, consideramos este trabalho extremamente pertinente como estudantes de engenharia informática para nos familiarizar com *Kubernetes* e com a automatização, instalação, configuração com *Ansible*, monitorização e avaliação duma aplicação, o que nos benificia muito no ramo *DevOps*. Assim, com o desenvolvimento deste trabalho sentimos que todos os conceitos estudados em Aplicações e Serviços de Computação em Nuvem foram fortemente consolidados e tiramos deste um balanço bastante positivo, percebendo que a realização dos guiões das aulas práticas revelou ser uma ferramenta fundamental na elaboração dos vários *playbooks* e na utilização da ferramenta *Ansible*.

Acreditamos que o trabalho poderia ser ainda melhorado pois não implementamos a avaliação experimental em grande escala, o que nos apresentaria os limites da solução implementada, e também não implementamos a escalabilidade e resiliência, que resolveria as limitações que encontraríamos na avaliação experimental em grande escala.