Universidade do Minho

Mestrado Integrado em Engenharia Informática

Departamento de Informática

System Deployment & Benchmarking

GitLab

António Sérgio Alves Costa A78296
José Pedro Moreira Resende A77486
Maria de La Salete Dias Teixeira A75281

9 de Novembro de 2018

Conteúdo

1 Introdução

No âmbito da UC de System Deployment & Benchemarking, foi-nos proposta a realização de um trabalho que consiste na automatização do processo de *deployment* e de *benchmarking* para uma aplicação à escolha pelo grupo.

A primeira fase do projeto baseou-se na seleção da aplicação e na sua análise pormenorizada, com a finalidade de determinar o padrão de distribuição mais adequado e eficiente.

Tendo isto em mente e considerando os requisitos exigidos pelos docentes na escolha da aplicação, o grupo decidiu optar pelo *GitLab*, uma plataforma online para gestão de repositórios *GIT*, equipa e *DevOps*.

2 Arquitetura e Componentes

Para tirar partido do *GitLab*, a sua arquitetura lógica tem de conter 4 camadas essenciais. No mais baixo nível tem-se a camada de base de dados, responsável por armazenar os dados dos utilizadores e respetivos repositórios. Acima desta está uma camada de cache, que comunica com a base de dados e armazena repostas a pedidos. De seguida, fica a camada da aplicação, onde também se encontram os repositórios *GIT*. Por último, é necessária uma camada de interface *web*, para os clientes poderem realizar pedidos e receber as respostas.

Como componente da base de dados, o *GitLab* pode usar o *MySQL* ou o *PostgreSQL*, tendo-se optado pelo segundo, e como componente para a cache utiliza-se o *Redis*. Para a interface *web* deve-se utilizar o *Nginx*, no entanto, o grupo optou por usar *Kubernetes* no *deployment* da aplicação, que por si já disponibiliza *loading balance*.

3 Padrões de Distribuição

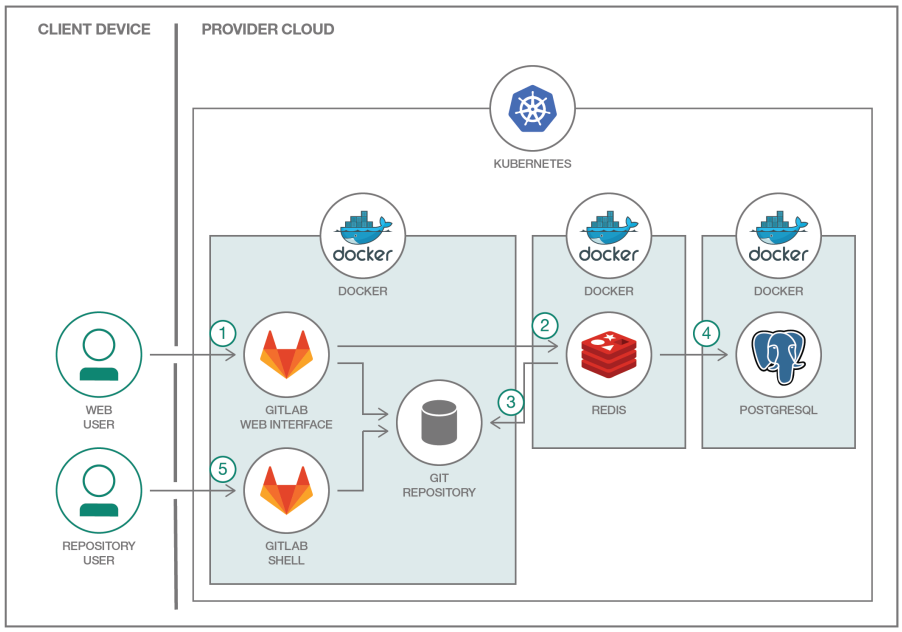

Figura 1: Arquitetura do sistema

A imagem acima representa o padrão de distribuição atual do projeto. Esta foi a arquitetura escolhida, visto que é facilmente implementável, especifica, a arquitetura usada pela própria *GitLab* e oferece bastantes vantagens que serão discutidas seguidamente.

Como se pode verificar, distribui-se os vários componentes por diferentes *containers*. Isto permite:

- **Isolamento dos serviços** Apenas se faz certo tipos de serviços em cada *container*, pois só se tem um componente por *container*.
- **Facilidade de deployment** Sempre que se quiser adicionar outro serviço, apenas é necessário criar outro *container* com esse serviço e configurar esse nodo para se enquadrar no resto da rede.
- **Escalabilidade** É possível replicar qualquer um dos serviços "infinitamente", ou seja, pode-se adicionar *containers* com os mesmos serviços quando tal seja necessário para escalar melhor a aplicação, ou simplesmente aumentar os recursos do *container* atual.
- **Portabilidade** Pode-se mover uma aplicação de um *host environment* para outro. O novo ambiente pode ser um novo sistema operativo, uma versão diferente do sistema operativo ou uma plataforma de hardware diferente. Esta mudança pode ser feita sem grandes esforços.

Apesar de tudo isto, pretende-se melhorar as configurações dos vário componentes, de maneira a otimizar a *perfomance* da aplicação. Como por exemplo, pensa-se utilizar uma*DBpool*, permitindo assim aproveitar conexões à base de dados, já abertas, recorrentemente.

4 Formas de Comunicação

Devido ao tipo de configuração que se tem com *Kubernetes*, cada componente existe isolado num *container* (sendo o runtime utilizado o *docker*) que esta contido no seu próprio *pod*, de maneira a que cada *pod* tem o seu ip e um espaço de portas divido por todos os *containers* que correm nesse *pod*. Todos estes *pods* comunicam entre si através de uma abstração de rede disponibilizada pelo *Kubernetes*, sendo que na nossa configuração é *Weave Net*, o que permite uma comunicação com latência reduzida. A nível aplicacional, os componentes vão comunicar entre si através de standard *Unix* sockets TPC/IP.

Para a utilização da interface *web* utilizamos http/https, isto vai incluir tudo o que seja operações ao nível aplicacional, como por exemplo a autenticação de um utilizador até ao clone de um repositório *GIT*.

5 Pontos de Configuração

Para dar *deploy* desta aplicação utilizou-se uma máquina *ubuntu* 16.04 com um *cluster Kubernetes*, tendo 1 *master* e 1 *worker* para cada um dos serviços com um *networking model Weave Net*.

São criados 3 *pods*, cada um com um *container docker* e o seu volume de persistência de dados de forma a correr cada um dos serviços isolados, isto é,

vamos ter um *pod* com *Redis*, um com *PostgreSQL* e outro com a aplicação *GitLab* a correr neste *cluster*.

6 Operações com Desempenho Crítico

Nesta aplicação, o objetivo será gerir repositórios *GIT*, logo tem que se garantir um alto desempenho e disponibilidade nas operações de gestão e consulta de um repositório, tanto pela interface *web* como pelos comandos *GIT* .

Desta forma, consideram-se operações criticas de desempenho, ao nível do ecossistema *GIT*, clonar um repositório e dar *push* de um novo *branch* para *upstream* ou de uma alteração num *branch* já em *upstream*, uma vez que estas são as operações mais básicas e essenciais no *workflow* de um sistema de controlo de versões.

Ao nível da interface *web*, tem-se como operações críticas o *login* de utilizadores, uma vez que será a operação mais comum a seguir a clonar um repositório, e a operação de consultar um repositório na interface *web*, que faz parte de todo o conceito de *social code*, permitindo descobrir novos projetos ou consultar projetos atuais.

7 Apreciação critica

Nesta primeira fase determinamos qual, na nossa opinião, o melhor padrão de distribuição para a aplicação em questão. Utilizamos *Kubernets*, pois é um standard para a implementação de *GitLab*, graças à existência de imagens de todos os componentes necessários para tal.

Apesar de ser um standard, tivemos vários entraves com *Kubernets*, devido a, para além de ser necessário aprender a utilizar este sistema de gestão de *containers*, foi necessário corrigir diversos erros. Ainda assim, continuámos com um problema, pois um dos volumes criados apaga-se e tem que se correr novamente o ficheiro de configuração para se ter todos os volumes corretamente instalados.

Em suma, concluímos esta fase com um padrão de distribuição definido e temos a aplicação a correr corretamente, apesar do problema na criação dos *containers*. Consideramos assim que esta primeira etapa foi concluída com sucesso e não só já temos bases para avançar para a etapa final, como já começámos a implementar algumas das otimizações especificadas anteriormente.

Referências

[1] GitLab Architecture Overview, https://docs.gitlab.com/ee/development/architecture.html#components, 24-10-2018

[2] Scaling the GitLab database, https://about.gitlab.com/2017/10/02/scaling-the-gitlab-database/, 31-10-2018