Trabalho Prático 3 - Computação em Nuvem - Tarefa 3

Aluno: Jean George Alves Evangelista

Matrícula: 2024661178

Descrição

Na tarefa 3 do trabalho foi um *runtime* conforme especificado no enunciado. O *runtime* estende as funcionalidades do *runtime* existente. Todos os requisitos da tarefa 3 foram implementados com sucesso.

Abordagem de Implementação

A tarefa 3 foi a mais difícil do trabalho. Inicialmente foi implementado um runtime que faz exatamente o que o *runtime* disponilizado na imagem docker *lucasmsp/serverless:redis* faz:

1. Carregar o método *handler* implementado na tarefa 1
2. De tempos em tempos, coletar dados do Redis e passá-los como argumento para a função *handler*, respeitando o padrão de parâmetros estabelecido
3. Salvar o retorno da função *handler* no Redis

O novo *runtime* seguia exatamente os mesmos padrões do *runtime* existente e utilizava o *ConfigMap pyfile*, que contém o código fonte da função *handler*. Um novo arquivo .yaml foi criado para implantar novamente a função *serverless* implementada na tarefa 1.

Uma vez verifcado que o novo *runtime* tinha o mesmo comportamento do antigo, modificações foram feitas na implementação para atender os requisitos pré-estabelecidos:

1. Permitir o usuário acessar uma chave diferente no Redis. Para isso, foi adicionada a variável de ambiente REDIS_INPUT_KEY, que é utilizada no código para determinar onde ler do Redis
2. Permitir que o usuário defina de quanto em quanto tempo, em segundos, os dados devem ser obtidos no Redis. Foi adicionada a variável de ambiente REDIS_MONITORING_PERIOD, que é utilizada no código.
3. Permitir que a *serverless* seja implementada em múltiplos arquivos .py. Foi criado um *ConfigMap*, que contém uma URL para um .zip no Google Drive:

https://drive.google.com/uc?id=1m3q6AxUC7plIWEPZqJXOScF5PTsVRD09&export=download. O arquivo zipado contém exatamente a mesma implementação da Tarefa 1, porém o código foi divido em múltimos arquivos. O novo *runtime* acessa a URL, faz o download do arquivo zipado e carrega os scripts Python para a memória, para que possam ser executados.

4. Permitir que o usuário defina um "ponto de entrada" da execução da função serverless. Foi implementado adicionando a variável de ambiente FUNCTION_HANDLER. O *runtime* tentará chamar a função com nome definido em FUNCTION_HANDLER.

Uma vez que as alterações foram realizadas no código e no arquivo de implantação, testes foram realizados para garantir que tudo estava funcionando conforme esperado:

1. Foi verificado que o *deployment* foi feito com sucesso e o *pod* estava funcionando sem erros
2. O novo *runtime* emite mensagens de alerta, erro e sucesso. Foi verificado que as mensagens corretas estavam sendo exibidas nos *logs* do *pod*.
3. Foi verificado que o dashboard implementado na tarefa 2 funcionava perfeitamente

Compatibilidade com o *runtime* fornecido

O novo runtime possui 100% de compatibilidade com o *runtime* fornecido. Ele faz exatamente o que o runtime fornecido faz, porém adiciona as funcionalidades extras descritas, permitindo maior flexibilidade. Cabe citar que se por qualquer motivo o processo de baixar o arquivo zipado e carregar os *scripts* Python em memória falhar, o novo *runtime* tentará seguir a implementação "antiga", isto é, ele irá executar o conteúdo do *pyfile*).