RUN.CODES

JOÃO BATISTA

JBATISTA@ICMC.USP.BR

1 Introdução

O Run.codes é um projeto de submissão e correção automática de programas, implementado por Felipe Simões Lage Gomes Duarte, Fábio Henrique Gomes Sikansi e Samuel Gomes Fadel, então alunos do ICMC-USP.

Este pequeno documento tem por objetivo mostrar os elementos constituintes do sistema, o seu funcionamento por meio de um diagrama de estados e a infraestrutura de hardware necessária para a configuração em múltiplos servidores.

O conteúdo aqui disponível foi resultado de algumas conversas com os desenvolvedores e da pouca documentação que encontramos da versão original. Juntamente com o manual de instalação disponível no GitHub, esperamos que seja, se não suficiente, um ponto de partida para uma melhor compreensão do sistema. Com isso, possibilitar a instalação e execução do RunCodes em um único servidor. Uma versão estendida deste manual será produzida em breve com detalhes da configuração e instalação em múltiplos servidores.

2 Infra estrutura e Diagrama de estados do Run.Codes

O Run.code roda em 3 máquinas distintas. Como utilizamos o serviço de nuvem do ICMC, as máquinas concedidas são virtuais (VMs), assim denominadas: a) VM WEB; b) VM BD e c) VM Compiler. Originalmente, o Run.codes utilizava um storage, contratado AWS (s3) para armazenamento do problemas (descrição), arquivos necessários para a execução dos problemas (*.in, *.out e arquivos adicionais).

Na migração para o ICMC, não usaremos storage externos. O armazenamento será parte de infraestrutura do Instituto. No nosso caso, o storage será parte integrante da VM BD. Exportaremos um diretório local da VM BD via SSHFS que será montado nas VM WEB e VM Compiler. O servico AWS foi substituído pelo SeaWeed que fornece uma API compatível.

A especificação das 3 VMs disponibilizadas pelo STI é exatamente aquela sugerida pelo desenvolvedor Felipe Lage, suficiente para manter o serviço em funcionamento de forma adequada para várias universidades do país. Como o uso agora restrito a apenas ao ICMC, pode-se assumir que a configuração será mais que suficiente para atender a demanda local.

A seguir, as especificações de cada VM, juntamente com dados que são importantes para o processo de instalação:

1. VM WEB - Frontend:

- Host: run-codes-web
- CPU: 2 Cores
- RAM: 2GB RAM
- ARMAZENAMENTO: 15GB
- Função: Frontend Web
- IPv4: N.N.N.N
- IPv6: XXXXXXXX
- Usuario: runcodes
- Senha: *****

2. VM BD e Storage:

    - Host: run-codes-db
    - CPU: 2 Cores
    - RAM: 2GB RAM
    - ARMAZENAMENTO: 15GB
    - Função: Database
    - IPv4: N.N.N.N1
    - IPv6: XXXXXX
    - Usuario: runcodes
    - Senha: *****

3. VM Compiler:

    - Host: run-codes-comp
    - CPU: 2 Cores
    - RAM: 2GB RAM
    - ARMAZENAMENTO: 60GB
    - Função: Compilação
    - IPv4: N.N.N.N2
    - IPv6: XXXXXXX
    - Usuario: runcodes
    - Senha: *****

A VM WEB é o front/backend, escrito em PHP. A VM DB hospeda banco de dados postgres e uma área de storage distinta. A VM Compiler é responsável pela compilação dos casos de teste. Por Banco de Dados entenda-se as informações gerenciais do Runcodes: usuários, universidades, disciplinas, etc. Já os casos de testes, majoritariamente compostos por arquivos texto estão nos storage. Casos de testes podem ser vários por exercícios e os tamanhos, cada qual podendo chegar a ordem de MB. Também ficam no storage as saídas produzidas pelos programas.

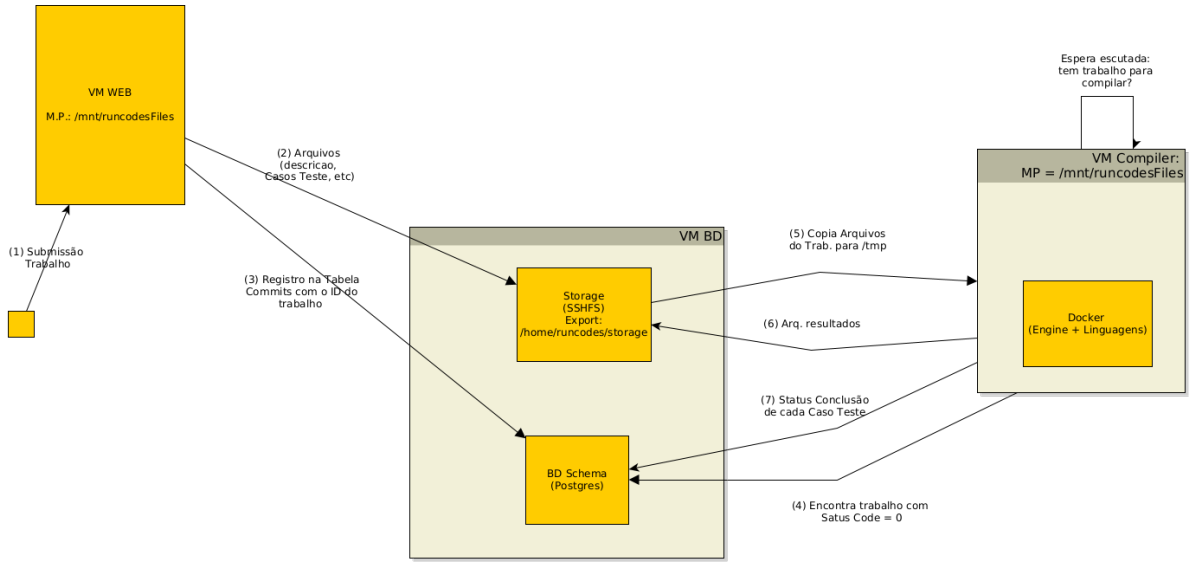

Figura 1: Diagrama de estados do Run.Codes

2.1 Funcionamento do Run.Codes

O diagrama da Figura 1 ilustra o fluxo de transações entre as 3 VMs descritas. Da submissão de um programa ao retorno dos resultados, temos as seguintes ações:

1. Usuário submete um trabalho
2. A VM WEB armazena no storage os arquivos referentes ao trabalho: descrição pdf, casos de teste, arquivo opcional para execução, etc; Um problema entrante no BD (ainda não executado) recebe STATUS CODE = 0.
3. A VM WEB coloca na Tabela Commits um ID para o trabalho: O BD postgres armazena somente informações necessárias para identificar e gerenciar o trabalho (ID trabalho, ID Aluno, etc). Os dados 'grossos' ficam no storage. **NOTA**: A VM Compiler fica em espera escutada consultando o BD por trabalhos a compilar;
4. Ao encontrar um registro com STATUS CODE = 0, pega o ID do arquivo no BD;
5. VM Compiler copia os arquivos do storage para uma pasta compartilhada /tmp. Faz o seu trabalho, executando o código e gerando as saídas;
6. Devolve para o storage os resultados do trabalho (*.in, *.out, *.err);
7. Registra no BD o status da execução para cada caso de teste. O BD possui uma tabela para isso. NOTA: A VM WEB fica consultando o storage para saber se já retornou o resultado do trabalho;
8. Retorna à VM Web sinalização de trabalho concluído;
9. VM WEB notifica usuário.